

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по Лабораторной Работе № 2

по дисциплине **«Проектирование и реализация баз данных»**

Вариант 12

Автор: Мазеин Никита Олегович

Факультет: ФИКТ

Группа: K32402

Преподаватель: Говорова Марина Михайловна



Санкт-Петербург

2023

1. Цель работы:

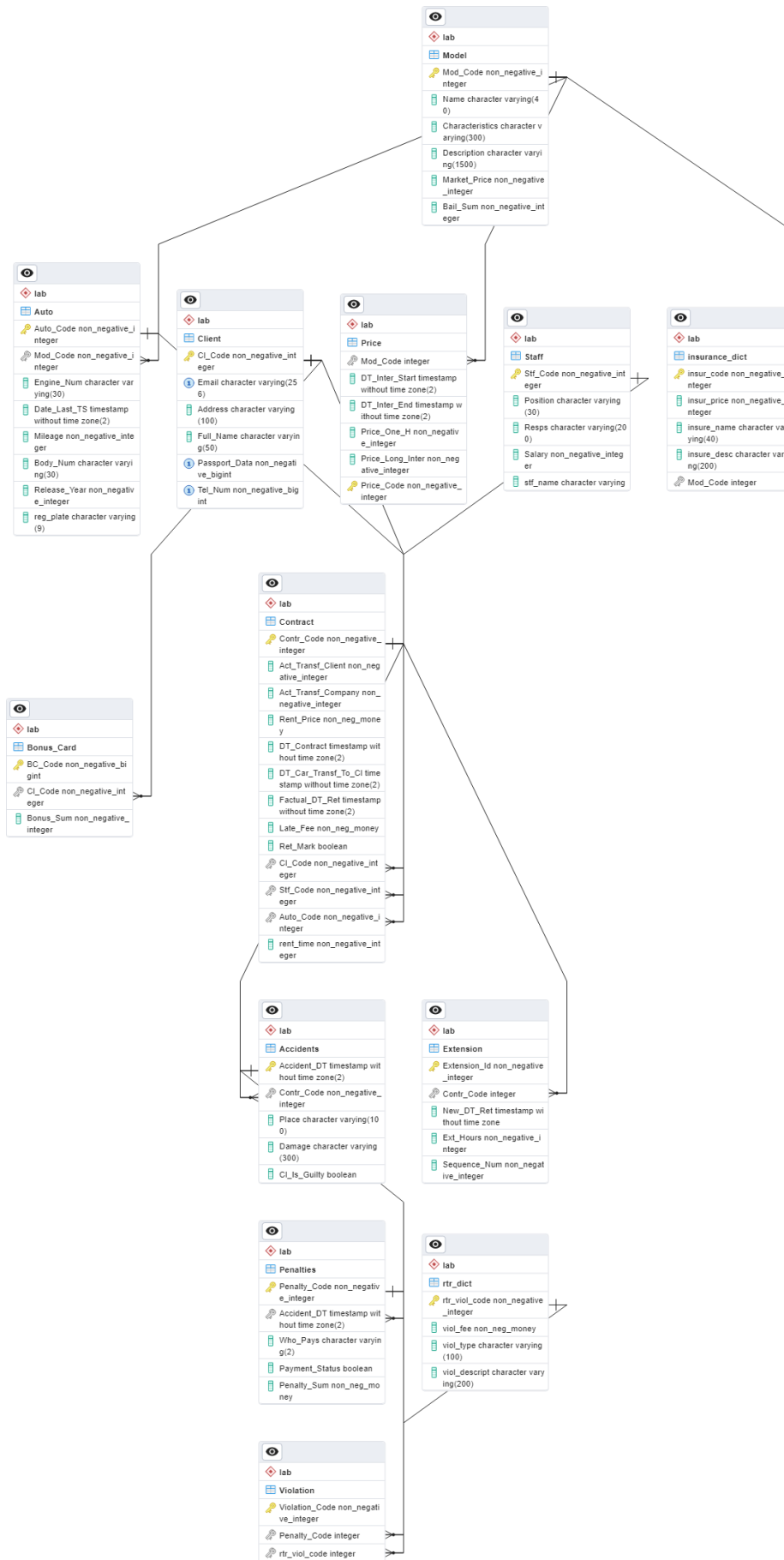
овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

2. Практическое задание:

БД «Прокат автомобилей»

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 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.

3. ERD Cxema:



4. Выполнение:

Задание 2. Создать запросы:

- Какой автомобиль находился в прокате максимальное количество часов?

```
1 SELECT "Auto_Code", rent_time
2 FROM lab."Contract"
3 WHERE rent_time =
4 (SELECT MAX(rent_time)
5 FROM lab."Contract")
```

| | Auto_Code integer | rent_time integer |
|---|----------------------|----------------------|
| 1 | 186549 | 72 |
| 2 | 776363 | 72 |

- Автомобили какой марки чаще всего брались в прокат?

```
1 SELECT m."Name", COUNT(*) AS rent_count
2 FROM lab."Model" m
3 JOIN lab."Auto" USING ("Mod_Code")
4 JOIN lab."Contract" USING ("Auto_Code")
5 GROUP BY m."Mod_Code"
6 HAVING COUNT(*) = (
7     SELECT MAX(rent_count)
8     FROM (
9         SELECT COUNT(*) AS rent_count
10        FROM lab."Model" m
11        JOIN lab."Auto" USING ("Mod_Code")
12        JOIN lab."Contract" USING ("Auto_Code")
13        GROUP BY m."Mod_Code"
14    ) AS subquery);
```

| | Name character varying (40) | rent_count bigint |
|---|--------------------------------|----------------------|
| 1 | BMW X5 | 9 |

- Определить убытки от простоя автомобилей за вчерашний день.

```

1 SELECT SUM(p.current_price * h.hours_wo_rent) AS total_loss
2 FROM (
3     SELECT a."Auto_Code",
4           (24 - LEAST(COALESCE(hours_rented, 0), 24)) AS hours_wo_rent
5 FROM lab."Auto" a
6 LEFT JOIN (
7     SELECT "Auto_Code",
8           COALESCE(CAST(SUM(EXTRACT(EPOCH FROM ("Factual_DT_Ret" - "DT_Car_Transf_To_Cl")) / 3600 AS INTEGER), 0)
9 FROM lab."Contract"
10 WHERE DATE("DT_Car_Transf_To_Cl") = DATE(CURRENT_DATE - 3)
11 GROUP BY "Auto_Code"
12 ) AS c ON a."Auto_Code" = c."Auto_Code"
13 ORDER BY a."Auto_Code"
14 ) AS h
15 JOIN (
16     SELECT a."Auto_Code", p."Price_One_H" AS current_price
17 FROM lab."Auto" a
18 JOIN lab."Price" p ON a."Mod_Code" = p."Mod_Code"
19 WHERE p."DT_Inter_End" IS NULL
20 ORDER BY a."Auto_Code"
21 ) AS p ON h."Auto_Code" = p."Auto_Code";

```

Data Output Messages Notifications



| | total_loss bigint |
|---|----------------------|
| 1 | 53504 |

- Вывести данные автомобиля, имеющего максимальный пробег.

```

1 SELECT "Auto_Code"
2 FROM lab."Auto"
3 WHERE "Mileage" = (
4     SELECT MAX("Mileage")
5     FROM lab."Auto");

```

Data Output Messages Notifications



| | Auto_Code [PK] integer |
|---|---------------------------|
| 1 | 539481 |

- Какой автомобиль суммарно находился в прокате дольше всех.

```

1 SELECT lab."Auto"."Auto_Code", SUM("rent_time")
2 FROM lab."Auto"
3 JOIN lab."Contract" ON lab."Auto"."Auto_Code" = lab."Contract"."Auto_Code"
4 GROUP BY lab."Auto"."Auto_Code"
5 HAVING SUM("rent_time") = (
6     SELECT MAX(total_rent_time)
7     FROM (
8         SELECT "Auto_Code", SUM("rent_time") AS total_rent_time
9         FROM lab."Contract"
10        GROUP BY "Auto_Code"
11     ) AS subquery
12 );
13

```

Data Output Messages Notifications



| | Auto_Code [PK] integer | sum bigint |
|---|---------------------------|---------------|
| 1 | 186549 | 168 |

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

```

1 SELECT m."Name", COUNT(*)
2 FROM lab."Auto" a
3 JOIN lab."Model" m ON a."Mod_Code" = m."Mod_Code"
4 GROUP BY m."Name";
5

```

Data Output Messages Notifications

| | Name character varying (40) | count bigint |
|---|--------------------------------|-----------------|
| 1 | Audi Q7 | 3 |
| 2 | Porsche Panamera | 3 |
| 3 | Audi A6 | 3 |
| 4 | Lexus LS | 4 |
| 5 | Jaguar F-Type | 3 |
| 6 | Mercedes-Benz E-Class | 6 |
| 7 | BMW X5 | 7 |
| 8 | Tesla Model S | 3 |

- Определить средний “возраст” автомобилей компании.

```

1 SELECT ROUND(AVG(EXTRACT(YEAR FROM CURRENT_DATE) - "Release_Year"), 3) AS avg_age
2 FROM lab."Auto";
3

```

Data Output Messages Notifications

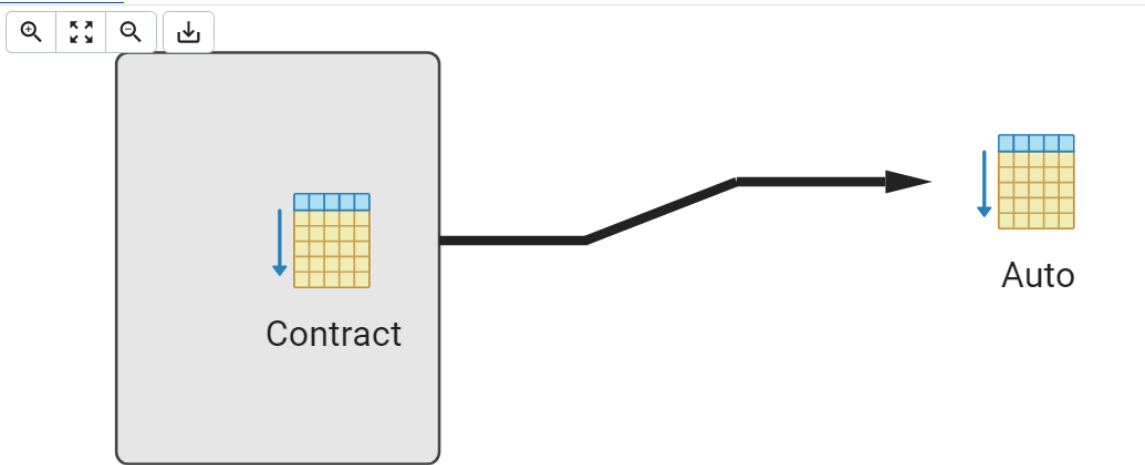
| | avg_age numeric |
|---|--------------------|
| 1 | 5.313 |

Задание 3. Создать представление:

- Какой автомобиль ни разу не был в прокате?

```
1 SELECT "Auto_Code" FROM lab."Auto"
2 WHERE "Auto_Code" NOT IN
3 (SELECT "Auto_Code"
4 FROM lab."Contract")
```

| | Auto_Code [PK] integer |
|----|---------------------------|
| 1 | 205635 |
| 2 | 759321 |
| 3 | 324516 |
| 4 | 362718 |
| 5 | 237389 |
| 6 | 184729 |
| 7 | 837287 |
| 8 | 276172 |
| 9 | 367932 |
| 10 | 452145 |
| 11 | 872364 |
| 12 | 539481 |
| 13 | 287461 |
| 14 | 721930 |
| 15 | 972841 |



| # | Node | Rows | |
|----|--|--------|-------|
| | | Actual | Loops |
| 1. | → Seq Scan on Auto as Auto (rows=15 loops=1) Filter: (NOT (hashed SubPlan 1)) Rows Removed by Filter: 17 | 15 | 1 |
| 2. | → Seq Scan on Contract as Contract (rows=33 loops=1) | 33 | 1 |

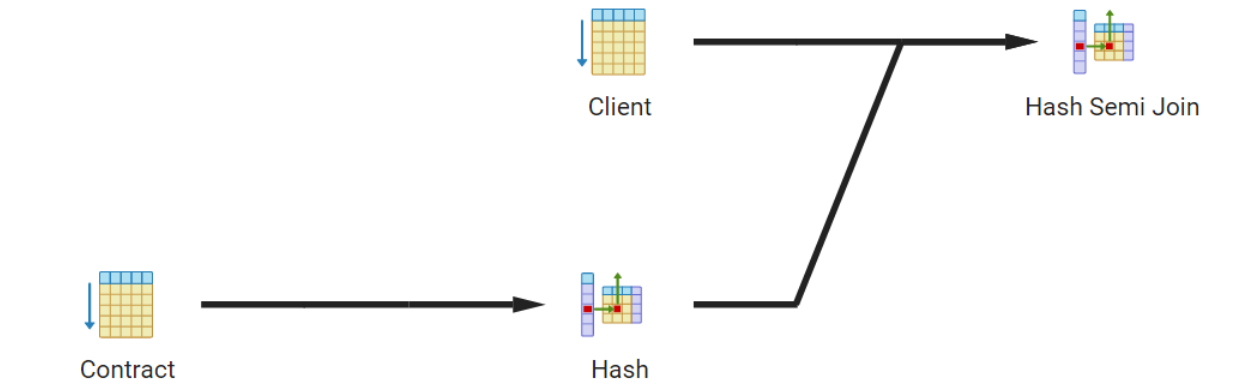
- Вывести данные клиентов, не вернувших автомобиль вовремя.

```

1 SELECT * FROM lab."Client"
2 WHERE "Cl_Code" IN
3 (SELECT "Cl_Code"
4 FROM lab."Contract"
5 WHERE "Late_Fee" IS NOT NULL);

```

| | Cl_Code [PK] integer | Email character varying (256) | Address character varying (100) | Full_Name character varying (50) | Passport_Data bigint | Tel_Num bigint |
|---|-------------------------|----------------------------------|------------------------------------|-------------------------------------|-------------------------|-------------------|
| 1 | 23456 | jane.smith@yahoo.com | 456 Elm St, Anytown, USA | Jane Smith | 2345678901 | 71234567891 |
| 2 | 34567 | bob.jones@hotmail.com | 789 Maple Ave, Anytown, USA | Bob Jones | 3456789012 | 71234567892 |
| 3 | 56789 | jim.smith@gmail.com | 456 Cedar Ave, Anytown, USA | Jim Smith | 5678901234 | 71234567894 |
| 4 | 78901 | mike.jones@hotmail.com | 123 Elm St, Anytown, USA | Mike Jones | 7890123456 | 71234567896 |
| 5 | 34568 | amy.brown@hotmail.com | 321 Maple Ave, Anytown, USA | Amy Brown | 6666666666 | 71234567917 |
| 6 | 11234 | peter.parker@hotmail.com | 444 Main St, New York City | Peter Parker | 6789678967 | 71234567971 |



| # | Node | Rows | |
|----|---|--------|-------|
| | | Actual | Loops |
| 1. | → Hash Semi Join (rows=6 loops=1) Hash Cond: (("Client"."Cl_Code")::integer = ("Contract"."Cl_Code")::integer) | 6 | 1 |
| 2. | → Seq Scan on Client as Client (rows=27 loops=1) | 27 | 1 |
| 3. | → Hash (rows=6 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB | 6 | 1 |
| 4. | → Seq Scan on Contract as Contract (rows=6 loops=1) Filter: ("Late_Fee" IS NOT NULL) Rows Removed by Filter: 27 | 6 | 1 |

Задание ?. Запросы на модификацию данных

- INSERT

```
1 INSERT INTO lab."Staff" ("Stf_Code", "Position", "Resps", "Salary", "stf_name")
2 SELECT subquery.next_code, subquery."Position", subquery."Resps", subquery."Salary", 'Johnny Depth'
3 FROM (
4     SELECT MAX("Stf_Code") + 1 AS next_code, "Position", "Resps", "Salary"
5     FROM lab."Staff"
6     WHERE "Position" IN (
7         SELECT "Position"
8         FROM lab."Staff"
9         GROUP BY "Position"
10        HAVING COUNT(*) = 1
11    ) AND "Salary" = (
12        SELECT MIN("Salary")
13        FROM lab."Staff"
14        WHERE "Position" IN (
15            SELECT "Position"
16            FROM lab."Staff"
17            GROUP BY "Position"
18            HAVING COUNT(*) = 1
19        )
20    )
21    GROUP BY "Position", "Resps", "Salary"
22 ) AS subquery;
```

🔍

🔍

🔍

📄

```
graph LR
    S1[Staff] --> A1[Aggregate]
    S1 --> H1[Hash]
    S1 --> A2[Aggregate]
    H1 --> H2[Hash Inner Join]
    A1 --> H2
    S2[Staff] --> N[Nested Loop Inner Join]
    A2 --> N
    N --> S[Sort]
    H2 --> A3[Aggregate]
    S --> A3
    A3 --> AS[SubQuery Scan]
    AS --> I[Insert]
```

| | | | | | |
|----|--------|------------------------------|---|-------|--------------|
| 11 | 678902 | Customer Serv Representative | Assists customers with inquiries and issues. Provides customer support. | 35000 | Johnny Depth |
|----|--------|------------------------------|---|-------|--------------|

- UPDATE

```
1 UPDATE lab."Auto"
2 SET "Date_Last_TS" = CURRENT_DATE
3 WHERE EXTRACT(YEAR FROM "Date_Last_TS") <= EXTRACT(YEAR FROM CURRENT_DATE) - 5;
```

Data Output Messages Explain × Notifications

UPDATE 9

Query returned successfully in 107 msec.

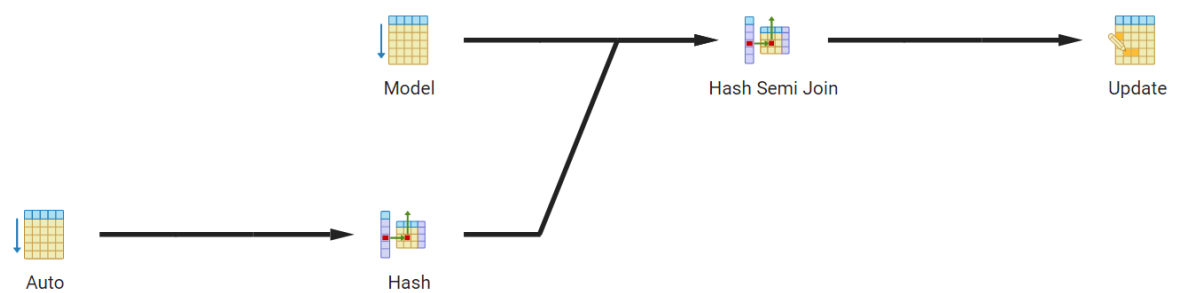


| Query | Query History |
|-------|--|
| 1 | UPDATE lab."Model" |
| 2 | SET "Bail_Sum" = "Bail_Sum" + 100 |
| 3 | WHERE "Mod_Code" IN (|
| 4 | SELECT "Mod_Code" |
| 5 | FROM lab."Auto" |
| 6 | WHERE "Date_Last_TS" = CURRENT_DATE); |

| Data Output | Messages | Explain | × | Notifications |
|-------------|----------|---------|---|---------------|
|-------------|----------|---------|---|---------------|

UPDATE 5

Query returned successfully in 77 msec.



- DELETE

Query Query History

1

DELETE FROM lab."Price"

2

WHERE "Mod_Code" IN (

3

SELECT "Mod_Code"

4

FROM lab."Auto"

5

WHERE "Auto_Code" NOT IN (

6

SELECT DISTINCT "Auto_Code"

7

FROM lab."Contract"

8

)

9

) AND "Price_One_H" <

10

(SELECT AVG("Price_One_H")

11

FROM lab."Price")

Data Output Messages Explain × Notifications

DELETE 601

Query returned successfully in 120 msec.



Задание ?. Создание индексов

Создадим простые индексы:

Используем код из предыдущего задания:

```
1 SELECT SUM(p.current_price * h.hours_wo_rent) AS total_loss
2 FROM (
3 SELECT a."Auto_Code",
4 (24 - LEAST(COALESCE(hours_rented, 0), 24)) AS hours_wo_rent
5 FROM lab."Auto" a
6 LEFT JOIN (
7 SELECT "Auto_Code",
8 COALESCE(CAST(SUM(EXTRACT(EPOCH FROM ("Factual_DT_Ret" - "DT_Car_Transf_To_Cl")) / 3600 AS INTEGER), 0) AS hours_rented
9 FROM lab."Contract"
10 WHERE DATE("DT_Car_Transf_To_Cl") = DATE(CURRENT_DATE - 3)
11 GROUP BY "Auto_Code"
12 ) AS c ON a."Auto_Code" = c."Auto_Code"
13 ORDER BY a."Auto_Code"
14 ) AS h
15 JOIN (
16 SELECT a."Auto_Code", p."Price_One_H" AS current_price
17 FROM lab."Auto" a
18 JOIN lab."Price" p ON a."Mod_Code" = p."Mod_Code"
19 WHERE p."DT_Inter_End" IS NULL
20 ORDER BY a."Auto_Code"
21 ) AS p ON h."Auto_Code" = p."Auto_Code";
```

Data Output Messages Notifications

Successfully run. Total query runtime: 78 msec.
1 rows affected.

Без Индексации: время выполнения 78 мс

```
1 CREATE INDEX idx_auto_auto_code ON lab."Auto" ("Auto_Code");
2
3 CREATE INDEX idx_contract_auto_code ON lab."Contract" ("Auto_Code");
4 CREATE INDEX idx_contract_dt_car_transf_to_cl ON lab."Contract" ("DT_Car_Transf_To_Cl");
5 CREATE INDEX idx_contract_factual_dt_ret ON lab."Contract" ("Factual_DT_Ret");
6
7 CREATE INDEX idx_price_mod_code ON lab."Price" ("Mod_Code");
8 CREATE INDEX idx_price_dt_inter_end ON lab."Price" ("DT_Inter_End");
```

Query Query History

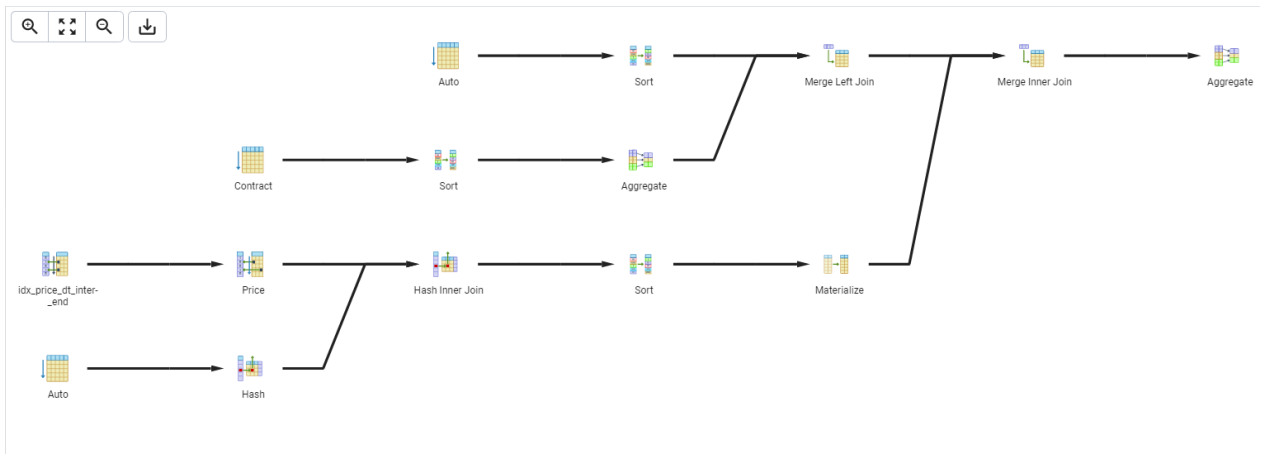
```
1 SELECT SUM(p.current_price * h.hours_wo_rent) AS total_loss
2 FROM (
3 SELECT a."Auto_Code",
4 (24 - LEAST(COALESCE(hours_rented, 0), 24)) AS hours_wo_rent
5 FROM lab."Auto" a
6 LEFT JOIN (
7 SELECT "Auto_Code",
8 COALESCE(CAST(SUM(EXTRACT(EPOCH FROM ("Factual_DT_Ret" - "DT_Car_Transf_To_Cl")) / 3600 AS INTEGER), 0) AS hours_rented
9 FROM lab."Contract"
10 WHERE DATE("DT_Car_Transf_To_Cl") = DATE(CURRENT_DATE - 3)
11 GROUP BY "Auto_Code"
```

Data Output Messages Notifications

Successfully run. Total query runtime: 46 msec.
1 rows affected.

После индексации: время выполнения 46 мс

Итог: при помощи индексации мы ускорили время выполнения запроса на 32мс (~25%)



| Graphical Analysis Statistics | | | | |
|-------------------------------|---|-------------|-------|---|
| # | Node | Rows Actual | Loops | |
| 1. | → Aggregate (rows=1 loops=1) | | 1 | 1 |
| 2. | → Merge Inner Join (rows=32 loops=1) | | 32 | 1 |
| 3. | → Merge Left Join (rows=32 loops=1) | | 32 | 1 |
| 4. | → Sort (rows=32 loops=1) | | 32 | 1 |
| 5. | → Seq Scan on Auto as a (rows=32 loops=1) | | 32 | 1 |
| 6. | → Aggregate (rows=0 loops=1) | | 0 | 1 |
| 7. | → Sort (rows=0 loops=1) | | 0 | 1 |
| 8. | → Seq Scan on Contract as Contract (rows=0 loops=1) Filter: (date("DT_Car_Transf_To_Cl") < (CURRENT_DATE - 3)) Rows Removed by Filter: 33 | | 0 | 1 |
| 9. | → Materialize (rows=32 loops=1) | | 32 | 1 |
| 10. | → Sort (rows=32 loops=1) | | 32 | 1 |
| 11. | → Hash Inner Join (rows=32 loops=1) Hash Cond: (p."Mod_Code" = (a_1."Mod_Code"); integer) | | 32 | 1 |
| 12. | → Bitmap Heap Scan on Price as p (rows=8 loops=1) Recheck Cond: ("DT_Inter_End" IS NULL) Heap Blocks: exact=1 | | 8 | 1 |
| 13. | → Bitmap Index Scan using idx_price_dt_inter_end (rows=8 loops=1) Index Cond: ("DT_Inter_End" IS NULL) | | 8 | 1 |
| 14. | → Hash (rows=32 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 10 kB | | 32 | 1 |
| 15. | → Seq Scan on Auto as a_1 (rows=32 loops=1) | | 32 | 1 |

Создадим составные индексы:

```

1 CREATE INDEX idx_time_in_rent
2 ON lab."Contract" ("Factual_DT_Ret", "DT_Car_Transf_To_Cl");

```

Successfully run. Total query runtime: 42 msec.
1 rows affected.

Итог: благодаря добавлению составного индекса удалось добиться ускорения выполнения запроса на 4 мс (1.8%)

Удалим Индексы:

| | | |
|----|----------|----------------------------------|
| 17 | Contract | idx_contract_auto_code |
| 18 | Contract | idx_contract_dt_car_transf_to_cl |
| 19 | Contract | idx_contract_factual_dt_ret |
| 20 | Price | idx_price_mod_code |
| 21 | Price | idx_price_dt_inter_end |
| 22 | Contract | idx_time_in_rent |

```

1 DROP INDEX lab."idx_time_in_rent";
2 DROP INDEX lab."idx_auto_auto_code";
3 DROP INDEX lab."idx_contract_auto_code";
4 DROP INDEX lab."idx_contract_dt_car_transf_to_cl";
5 DROP INDEX lab."idx_contract_factual_dt_ret";
6 DROP INDEX lab."idx_price_mod_code";
7 DROP INDEX lab."idx_price_dt_inter_end";
8 DROP INDEX lab."idx_contract_auto_code";

```

DROP INDEX

Query returned successfully in 69 msec.

| | tablename name | indexname name |
|----|--------------------------|--------------------------|
| 1 | Accidents | Accidents_pkey |
| 2 | Auto | Auto_pkey |
| 3 | Bonus_Card | Bonus_Card_pkey |
| 4 | Client | Client_pkey |
| 5 | Contract | Contract_pkey |
| 6 | Extension | Extension_pkey |
| 7 | Model | Model_pkey |
| 8 | Penalties | Penalties_pkey |
| 9 | Price | Price_pkey |
| 10 | Staff | Staff_pkey |
| 11 | Violation | Violation_pkey |
| 12 | insurance_dict | insurance_dict_pkey |
| 13 | rtr_dict | rtr_dict_pkey |
| 14 | Client | unq_email |
| 15 | Client | unq_pspst |
| 16 | Client | unq_telnum |

Все индексы удалены.

5. Выводы:

По итогам выполнения лабораторной работы были выполнены SELECT запросы к созданной базе данных, 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.