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»**

Автор: Гнеушев В. А.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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 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.

Описание проекта

В этом отчете я буду описывать запросы к спроектированной и созданной мной на работе базе данных.

Система, для которой создавалась база данных – это обучающая платформа, интегрированная в Telegram бота. У пользователей платформы есть 3 роли – ученик, учитель и администратор.

Ученики получают доступ к образовательной платформе, оплатив обучение и авторизовавшись в Telegram боте при помощи почты, указанной при оплате.

К отчету прилагается модель .”erwin” базы данных, с которой я работал.

Описание запросов

Получение списка кураторов (учителей) с количеством студентов, которых он обучает:

```
select
"curator".*,
COUNT("user_curator"."id") AS "students_count"
from "curator"
left join "user_curator"
    on "curator"."id" = "user_curator"."curator_id"
group by "curator"."id"
```

Добавление телеграм-пользователя, с обновлением его данных в случае если пользователь уже есть в таблице:

```
insert into "telegram_user"
("id", "first_name",
"last_name", "username", "role")
values ($1, $2, $3, $4, $5)
on conflict ("id")
do update set
    "first_name" = $2,
    "last_name" = $3,
    "username" = $4,
    "role" = coalesce($5, "telegram_user"."role")
```

Поиск пользователей по имени/telegram id/юзернейму:

```
select tu.*
from telegram_user as tu
where
(
    (tu.first_name || coalesce(' ' || tu.last_name, ''))
        ~ ('.*' || $1 || '.*')
    or (tu.id = $1)
    or (tu.username ~ ('.*' || $1 || '.*'))
)
```

Получение статистики по успеваемости ученика и рейтинга ученика, основанного на его успеваемости:

```
with user_rating as (
    select
        coalesce(
            avg(uhr.rating)
            filter(where hw.is_rated = true), 0
        ) as rating_avg,
        coalesce(
            sum(uhr.rating)
            filter(where hw.is_rated = true), 0
        ) as rating,
        count(distinct hw.id) filter(where hw.is_homework = true)
            as total_homeworks_count,
        count(distinct lsn.id) filter(where hw.is_homework = true)
            as total_lessons_count,
        count(distinct mdl.id) filter(where hw.is_homework = true)
            as total_modules_count,
        row_number() over(
            order by coalesce(
                sum(uhr.rating)
                filter(where hw.is_rated = true), 0
            ) desc
        ) as rating_position,
        (
            select count(distinct ulf.lesson_id)
            from user_lesson_feedback ulf
```

```
            where ulf.telegram_user_id = tu.id
        ) as lesson_feedbacks_count,
        (
            select coalesce(avg(ulf.rating), 0)
            from user_lesson_feedback ulf
            where ulf.telegram_user_id = tu.id
        ) as avg_lesson_feedback,
        tu.id as telegram_user_id
    from telegram_user as tu
    left join user_homework_rating as uhr
        on tu.id = uhr.telegram_user_id
    left join homework as hw
        on hw.id = uhr.homework_id
    left join lesson as lsn
        on lsn.id = hw.lesson_id
    left join module as mdl
        on mdl.id = lsn.module_id
    where tu.role = 'student'
    group by tu.id
    order by rating_position
)
select
    *
from user_rating
where telegram_user_id = $1
```

Получение даты первой активности пользователя на сайте:

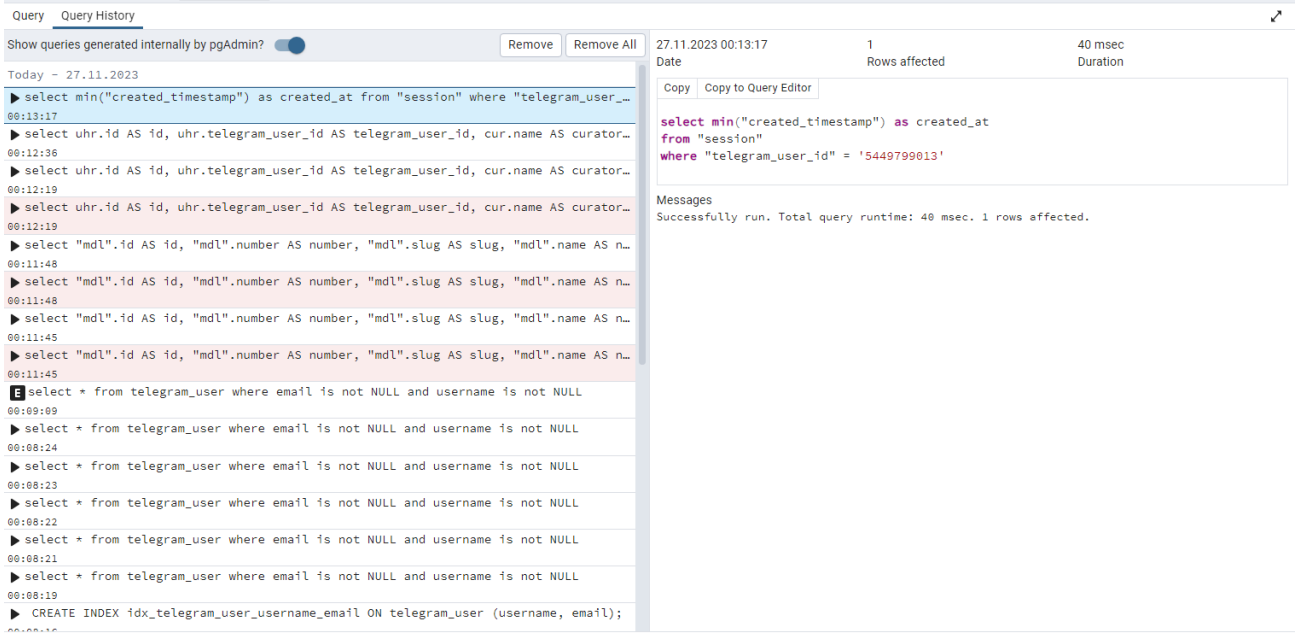
```
select min("created_timestamp") as created_at
from "session"
where "telegram_user_id" = $1
```

Получение структуры обучающего курса:

```
select
    "mdl".id AS id,
    "mdl".number AS number,
    "mdl".slug AS slug,
    "mdl".name AS name,
    "lsn".id AS lesson_id,
    "lsn".number AS lesson_number,
    "lsn".slug AS lesson_slug,
    "lsn".name AS lesson_name,
    "lsn".module_id AS lesson_module_id,
    "hw".id AS homework_id,
    "hw".number AS homework_number,
    "hw".slug AS homework_slug,
    "hw".name AS homework_name,
    "hw".lesson_id AS homework_lesson_id,
    "hw".is_bonus AS homework_is_bonus
from "module" as "mdl"
right join "lesson" as "lsn"
    on "lsn"."module_id" = "mdl"."id"
```

```
right join "homework" as "hw"
    on "hw"."lesson_id" = "lsn"."id"
where ($1::boolean is null or "hw".is_bonus = $1)
order by
    "mdl".number,
    "lsn".number,
    "hw".number;
```

Просмотр истории запросов:



Создание и проверка индексов

Запрос для тестирования:

```
1  select tu.first_name, s.created_timestamp as website_login_time
2  from telegram_user as tu
3  right join session as s
4      on s.telegram_user_id = tu.id
5  where
6      email is not NULL
7      and
8      username is not NUll
```

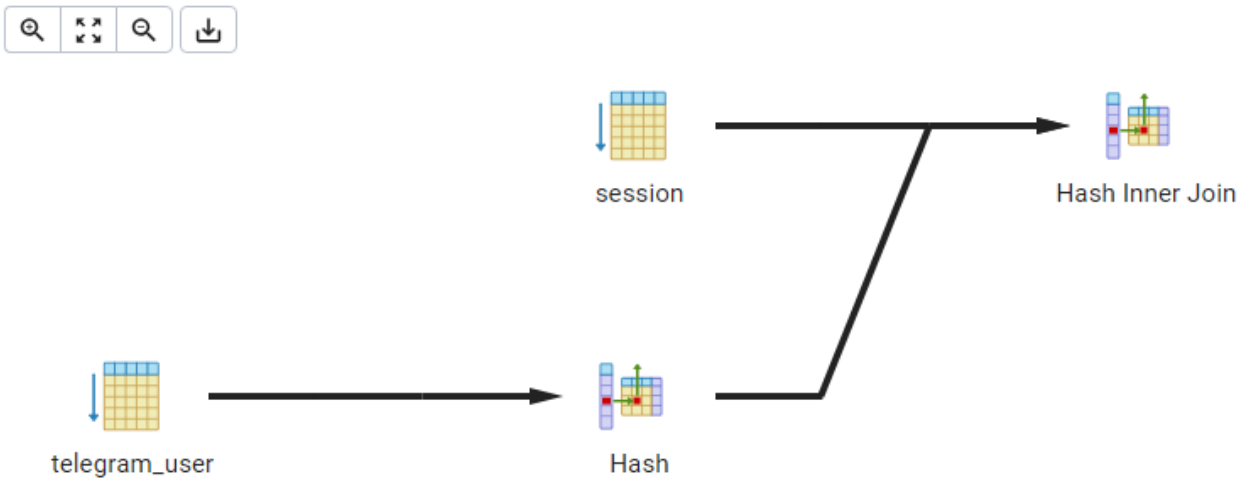
Время запроса без индекса:

```
Successfully run. Total query runtime: 94 msec.
532 rows affected.
```

Время запроса с индексом:

```
Successfully run. Total query runtime: 66 msec.
532 rows affected.
```

Графический анализ запроса:



Вывод

В ходе данной лабораторной работы я научился созданию индексов и проверил, насколько они понижают время выполнения запросов. Также узнал о существовании Query History и Explain в pgAdmin.