Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №1

По дисциплине «Распределённые системы хранения данных» (семестр 6)

Студент:
Дениченко Александр P3312
Практик:
Осипов Святослав

Санкт-Петербург
2025 г.

Задание

Используя сведения из системных каталогов, получить информацию обо всех файлах данных, доступных для чтения и записи. Полученную информацию представить в следующем формате:

```
No. FILE# CREATION_TIME STATUS
-- ----- -----------  --------------
1 00000001 2014-09-10 00:00:00 ONLINE
```

Выполнение

Листинг 1: script.sql

```
CREATE OR REPLACE PROCEDURE show_files() LANGUAGE plpgsql AS $$
DECLARE
    X record;
BEGIN

    RAISE NOTICE 'No. FILE#       NAME       MODIFICATION_TIME             SPACE';
    RAISE NOTICE '--- ----------- --------   --------------------------    ----------';

    FOR X IN (
        select
            ROW_NUMBER() OVER (ORDER BY(relfilenode)) as n,
            relfilenode as node,
            relname as rname,
            (pg_stat_file(pg_relation_filepath(pg_class.oid))).modification as modif,
            nspname as spac
        from pg_class
        join pg_namespace on pg_namespace.oid = pg_class.relnamespace
        where relkind = 'r' and nspname != 'pg_catalog' and nspname != 'information_schema'
    ) LOOP
        RAISE NOTICE '%    %    %    %    %',
            X.n::text,
            X.node::text,
            X.rname,
            to_char(X.modif::timestamp, 'YYYY-MM-DD HH24:MI:SS'),
            X.spac;
    END LOOP;
END;
$$;
```

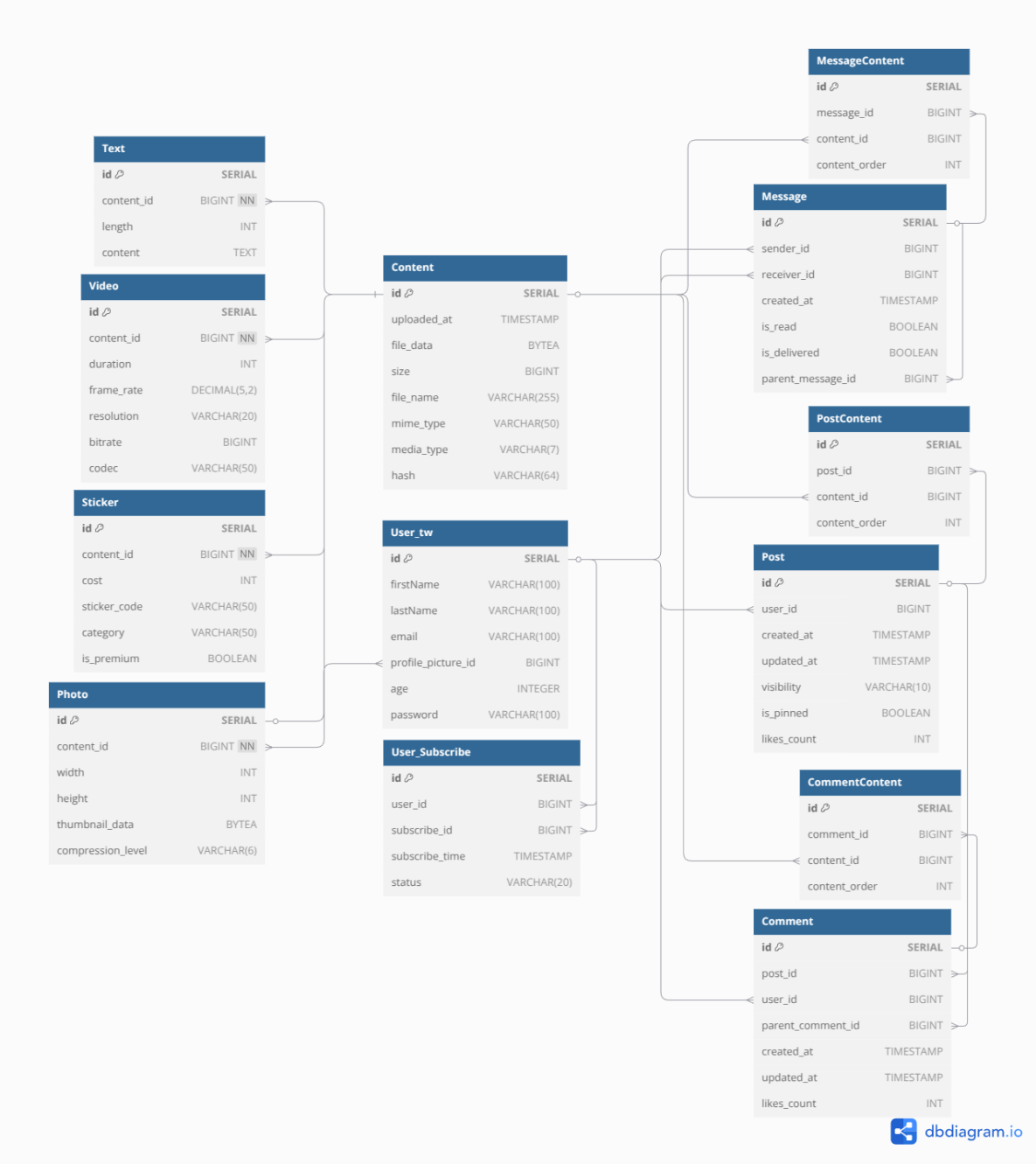
Листинг 2: kitty

```
postgres=# call show_files();
NOTICE:  No. FILE#       NAME       MODIFICATION_TIME             SPACE
NOTICE:  --- ----------- --------   --------------------------    ----------
NOTICE:  1    16389     users      2025-02-15 19:15:38     public
NOTICE:  2    16397     products     2025-02-15 19:15:38     public
NOTICE:  3    16455     u_order     2025-02-15 19:15:38     public
NOTICE:  4    16481     order_products     2025-02-15 19:15:38     public
NOTICE:  5    16497     data_directory     2025-02-19 10:16:39     public
CALL
```

Дополнительное задание

Сделать систему хранения данных схожую с facebook на postgres и на MongoDB, провести анализ быстродействия и удобства.

Postgres



Примеры запросов:

Задача: Найти 10 самых популярных элементов контента (фото, видео, текст, стикеры) за последний месяц, основываясь на общем количестве лайков постов, в которых они использовались. Также, для каждого элемента контента, вернуть имена и email создателей этих постов.

Сложность: Требует объединения множества таблиц (Content, PostContent, Post, User_tw) и агрегации данных. Сложно масштабировать при увеличении объема данных.

Листинг 3: TOP10

```
EXPLAIN ANALYZE
SELECT
    c.id AS content_id,
    c.media_type,
    c.file_name,
    SUM(p.likes_count) AS total_likes,
    STRING_AGG(DISTINCT u.firstName || ' ' || u.lastName, ', ') AS creator_names,
    STRING_AGG(DISTINCT u.email, ', ') AS creator_emails
FROM
    Content c
JOIN
    PostContent pc ON c.id = pc.content_id
JOIN
    Post p ON pc.post_id = p.id
JOIN
    User_tw u ON p.user_id = u.id
WHERE
    p.created_at >= NOW() - INTERVAL '1 month'
GROUP BY
    c.id, c.media_type, c.file_name
ORDER BY
    total_likes DESC
LIMIT 10;
```

Листинг 4: explain analyze without indexes

```
Planning Time: 0.232 ms
Execution Time: 206.047 ms
```

Листинг 5: explain analyze with indexes

```
Planning Time: 0.613 ms
Execution Time: 191.192 ms
```

Задача: Для заданного пользователя (например, с user_id = 123), найти 5 наиболее подходящих элементов контента, которые он еще не видел, основываясь на следующих критериях: У контента есть лайки. Контент от пользователей, на которых он подписан. В данном случае, будем считать, что "похожий"контент - это контент того же media_type.

Сложность: Требует объединения таблиц подписок, постов, контента и пользователей, а также логики для фильтрации просмотренного контента.

Листинг 6: recomend

```
EXPLAIN ANALYZE
WITH user_likes AS (
    SELECT DISTINCT c.media_type
    FROM Post p
    JOIN PostContent pc ON p.id = pc.post_id
    JOIN Content c ON pc.content_id = c.id
    WHERE p.user_id = 123 AND p.likes_count > 0
),
subscribed_users_content AS (
    SELECT c.id AS content_id, c.media_type, p.created_at
    FROM User_Subscribe us
    JOIN Post p ON us.subscribe_id = p.user_id
    JOIN PostContent pc ON p.id = pc.post_id
    JOIN Content c ON pc.content_id = c.id
    WHERE us.user_id = 123 AND us.status = 'approved'
```

```
    ),
    similar_content AS (
        SELECT c.id AS content_id, c.media_type, p.created_at
        FROM Post p
        JOIN PostContent pc ON p.id = pc.post_id
        JOIN Content c ON pc.content_id = c.id
        JOIN user_likes ul ON c.media_type = ul.media_type
        WHERE p.user_id <> 123
    )
    SELECT content_id, media_type
    FROM (
        SELECT content_id, media_type, created_at FROM subscribed_users_content
        UNION ALL
        SELECT content_id, media_type, created_at FROM similar_content
    ) AS combined_content
    WHERE content_id NOT IN (SELECT content_id FROM PostContent WHERE post_id IN (SELECT id
        FROM Post WHERE user_id = 123))
    ORDER BY created_at DESC
    LIMIT 5;
```

Листинг 7: explain analyze without indexes

```
    Planning Time: 0.459 ms
    Execution Time: 79.735 ms
```

Листинг 8: explain analyze with indexes

```
    Planning Time: 0.964 ms
    Execution Time: 55.243 ms
```

Задача: Найти всех пользователей, которые подписаны друг на друга (A подписан на B, и B подписан на A).
Сложность: Требует self-join и проверки условия взаимности.

Листинг 9: subs

```
    EXPLAIN ANALYZE
    SELECT
        us1.user_id AS user1_id,
        us2.user_id AS user2_id
    FROM
        User_Subscribe us1
    JOIN
        User_Subscribe us2 ON us1.user_id = us2.subscribe_id AND us1.subscribe_id =
            us2.user_id
    WHERE
        us1.user_id < us2.user_id
        AND us1.status = 'approved'
        AND us2.status = 'approved';
```

Листинг 10: explain analyze without indexes

```
    Planning Time: 0.099 ms
    Execution Time: 22.437 ms
```

Листинг 11: explain analyze with indexes

```
    Planning Time: 0.267 ms
    Execution Time: 16.065 ms
```