

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 Р3312

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

```
1 CREATE OR REPLACE PROCEDURE show_files() LANGUAGE plpgsql AS $$
2 DECLARE
3     X record;
4 BEGIN
5
6     RAISE NOTICE 'No. FILE# NAME MODIFICATION_TIME SPACE';
7     RAISE NOTICE '----' ;
8
9     FOR X IN (
10         select
11             ROW_NUMBER() OVER (ORDER BY(relfilenode)) as n,
12             relfilenode as node,
13             relname as rname,
14             (pg_stat_file(pg_relation_filepath(pg_class.oid))).modification as modif,
15             nspname as spac
16         from pg_class
17         join pg_namespace on pg_namespace.oid = pg_class.relnamespace
18         where relkind = 'r' and nspname != 'pg_catalog' and nspname != 'information_schema'
19     ) LOOP
20         RAISE NOTICE '% % % % %',
21             X.n::text,
22             X.node::text,
23             X.rname,
24             to_char(X.modif::timestamp, 'YYYY-MM-DD HH24:MI:SS'),
25             X.spac;
26     END LOOP;
27 END;
28 $$;
```

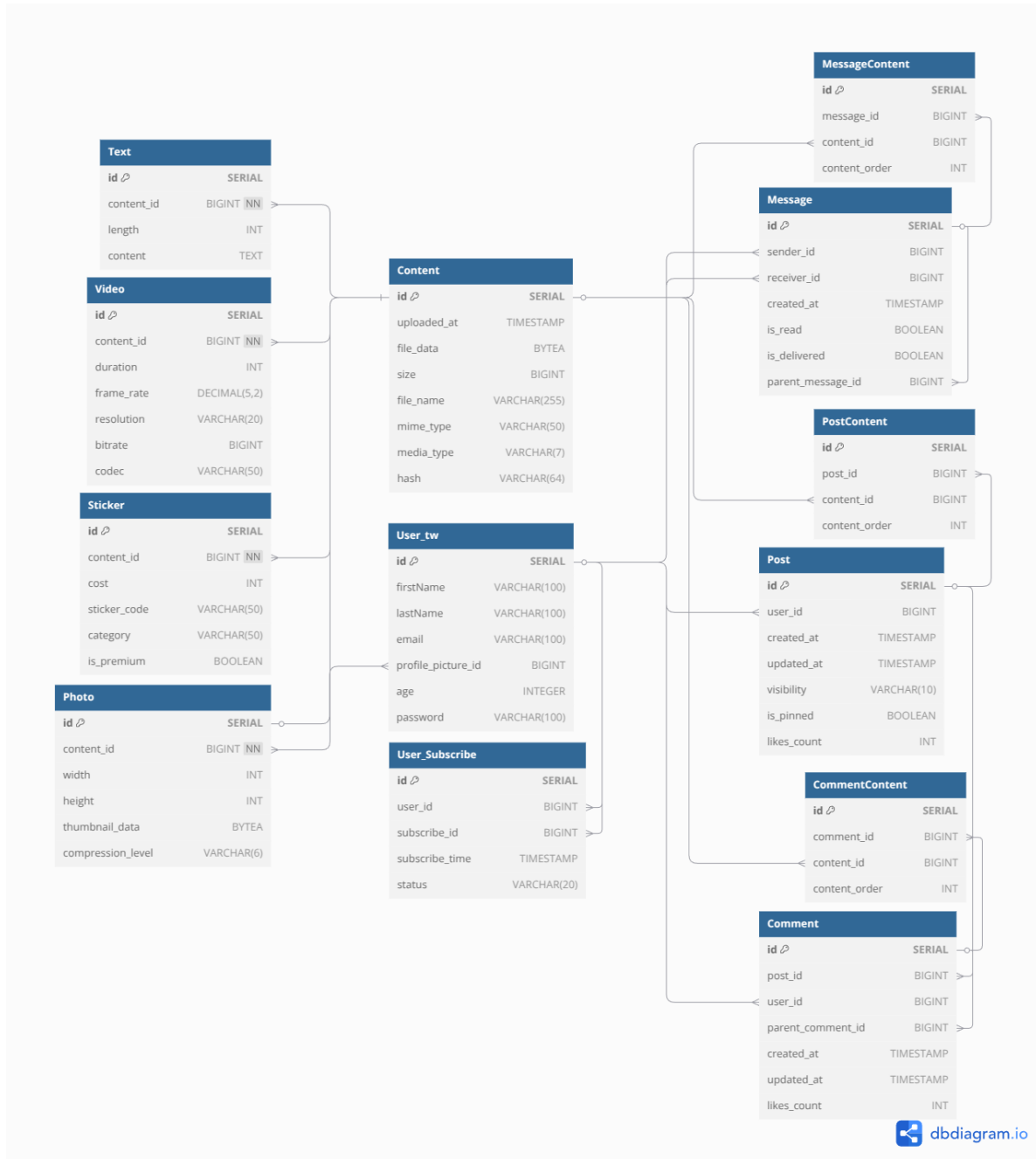
Листинг 2: kitty

```
1 postgres=# call show_files();
2 NOTICE: No. FILE# NAME MODIFICATION_TIME SPACE
3 NOTICE: ----
4 NOTICE: 1 16389 users 2025-02-15 19:15:38 public
5 NOTICE: 2 16397 products 2025-02-15 19:15:38 public
6 NOTICE: 3 16455 u_order 2025-02-15 19:15:38 public
7 NOTICE: 4 16481 order_products 2025-02-15 19:15:38 public
8 NOTICE: 5 16497 data_directory 2025-02-19 10:16:39 public
9 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: Требуется объединения множества таблиц (Content, PostContent, Post, User_tw) и агрегации данных. Сложно масштабировать при увеличении объема данных.

Листинг 3: TOP10

```

1  EXPLAIN ANALYZE
2  SELECT
3      c.id AS content_id,
4      c.media_type,
5      c.file_name,
6      SUM(p.likes_count) AS total_likes,
7      STRING_AGG(DISTINCT u.firstName || ' ' || u.lastName, ', ') AS creator_names,
8      STRING_AGG(DISTINCT u.email, ', ') AS creator_emails
9  FROM
10     Content c
11  JOIN
12     PostContent pc ON c.id = pc.content_id
13  JOIN
14     Post p ON pc.post_id = p.id
15  JOIN
16     User_tw u ON p.user_id = u.id
17  WHERE
18     p.created_at >= NOW() - INTERVAL '1 month'
19  GROUP BY
20     c.id, c.media_type, c.file_name
21  ORDER BY
22     total_likes DESC
23  LIMIT 10;

```

Листинг 4: explain analyze without indexes

```

1  Planning Time: 0.232 ms
2  Execution Time: 206.047 ms

```

Листинг 5: explain analyze with indexes

```

1  Planning Time: 0.613 ms
2  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: Требуется объединение таблиц подписок, постов, контента и пользователей, а также логики для фильтрации просмотренного контента.

Листинг 6: recommend

```

1  EXPLAIN ANALYZE
2  WITH user_likes AS (
3      SELECT DISTINCT c.media_type
4      FROM Post p
5      JOIN PostContent pc ON p.id = pc.post_id
6      JOIN Content c ON pc.content_id = c.id
7      WHERE p.user_id = 123 AND p.likes_count > 0
8  ),
9  subscribed_users_content AS (
10     SELECT c.id AS content_id, c.media_type, p.created_at
11     FROM User_Subscribe us
12     JOIN Post p ON us.subscribe_id = p.user_id
13     JOIN PostContent pc ON p.id = pc.post_id
14     JOIN Content c ON pc.content_id = c.id
15     WHERE us.user_id = 123 AND us.status = 'approved'

```

```

16  ),
17  similar_content AS (
18      SELECT c.id AS content_id, c.media_type, p.created_at
19      FROM Post p
20      JOIN PostContent pc ON p.id = pc.post_id
21      JOIN Content c ON pc.content_id = c.id
22      JOIN user_likes ul ON c.media_type = ul.media_type
23      WHERE p.user_id <> 123
24  )
25  SELECT content_id, media_type
26  FROM (
27      SELECT content_id, media_type, created_at FROM subscribed_users_content
28      UNION ALL
29      SELECT content_id, media_type, created_at FROM similar_content
30  ) AS combined_content
31  WHERE content_id NOT IN (SELECT content_id FROM PostContent WHERE post_id IN (SELECT id
32      FROM Post WHERE user_id = 123))
33  ORDER BY created_at DESC
34  LIMIT 5;

```

Листинг 7: explain analyze without indexes

```

1  Planning Time: 0.459 ms
2  Execution Time: 79.735 ms

```

Листинг 8: explain analyze with indexes

```

1  Planning Time: 0.964 ms
2  Execution Time: 55.243 ms

```

Задача: Найти всех пользователей, которые подписаны друг на друга (А подписан на В, и В подписан на А).
Сложность: Требуется self-join и проверки условия взаимности.

Листинг 9: subs

```

1  EXPLAIN ANALYZE
2  SELECT
3      us1.user_id AS user1_id,
4      us2.user_id AS user2_id
5  FROM
6      User_Subscribe us1
7  JOIN
8      User_Subscribe us2 ON us1.user_id = us2.subscribe_id AND us1.subscribe_id =
9      us2.user_id
10 WHERE
11     us1.user_id < us2.user_id
12     AND us1.status = 'approved'
13     AND us2.status = 'approved';

```

Листинг 10: explain analyze without indexes

```

1  Planning Time: 0.099 ms
2  Execution Time: 22.437 ms

```

Листинг 11: explain analyze with indexes

```

1  Planning Time: 0.267 ms
2  Execution Time: 16.065 ms

```

1 MongoDB

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.

Листинг 12: analyze without indexes

```
1 Query 1 (Top 10 popular): 0s 53.351318ms
```

Листинг 13: analyze with indexes

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`.

Листинг 14: analyze without indexes

```
1 Query 2 (Recommendations): 0s 261.533013ms
```

Листинг 15: analyze with indexes

Задача: Найти всех пользователей, которые подписаны друг на друга (А подписан на В, и В подписан на А).

Листинг 16: analyze without indexes

```
1 Query 3 (Mutual subs): 3s 69.803046ms
```

Листинг 17: analyze with indexes