САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе № 5

Дисциплина: «Базы данных»

Тема: «SQL-программирование: хранимые процедуры»

Выполнил студент гр. 43501/3 _____ А.Ю. Ламтев
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

«___» _____ 2019 г.

Санкт-Петербург
2019

Содержание

1. Цели работы

Познакомиться с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур.

2. Программа работы

1. Изучение возможностей языка `PL/pgSQL`.
2. Создание двух хранимых процедур в соответствии с индивидуальным заданием, полученным у преподавателя.
3. Выкладывание скрипта с созданными сущностями в репозиторий.
4. Демонстрация результатов преподавателю.

3. Хранимая процедура 1

Задание: На основании данных о подписках пользователя и о популярности фильмов по категориям сформировать список предложений для заданного пользователя.

В листинге 1 представлена процедура `up_to_n_movie_recommendations_for_user(n INTEGER, u_id BIGINT)`, которая формирует выборку в соответствии с заданием. Она принимает 2 аргумента: `n` — число предложений и `u_id` — идентификатор пользователя.

```
CREATE OR REPLACE FUNCTION up_to_n_user_favourite_categories(n INTEGER, id BIGINT)
  RETURNS TABLE
          (
            user_id          BIGINT,
            category_id      SMALLINT,
            max_movie_count BIGINT
          )
  LANGUAGE SQL AS
$$
WITH user_category_movie_count AS (
  WITH user_movie_category AS (
    SELECT s.user_id,
           mc.movie_id,
           mc.category_id
    FROM subscription_movie sm
           JOIN subscription s
                ON sm.subscription_id = s.id
           JOIN movie_category mc
                ON sm.movie_id = mc.movie_id
    GROUP BY s.user_id, mc.movie_id, mc.category_id
    )
    SELECT user_id,
           category_id,
           count(movie_id) AS movie_count
    FROM user_movie_category
    GROUP BY user_id, category_id
)
SELECT user_id,
       category_id,
       max(movie_count) AS max_movie_count
FROM user_category_movie_count
WHERE user_id = id
GROUP BY user_id, category_id
ORDER BY max_movie_count DESC
```

```sql
LIMIT n
$$;


CREATE OR REPLACE FUNCTION categories_with_maximum_audience()
  RETURNS TABLE
          (
            category_id SMALLINT,
            movie_id    INTEGER,
            audience    BIGINT
          )
  LANGUAGE SQL AS
$$
SELECT mc.category_id,
       mc.movie_id,
       count(DISTINCT s.user_id) AS audience
FROM movie_category mc
       JOIN subscription_movie sm
            ON mc.movie_id = sm.movie_id
       JOIN movie m
            ON sm.movie_id = m.id
       JOIN subscription s
            ON sm.subscription_id = s.id
WHERE m.series_season_id IS NULL
GROUP BY mc.category_id, mc.movie_id
ORDER BY audience DESC
$$;


CREATE OR REPLACE FUNCTION up_to_n_movie_recommendations_for_user(n INTEGER, u_id BIGINT)
  RETURNS TABLE
          (
            movie_id    INTEGER,
            category_id SMALLINT
          )
  LANGUAGE SQL AS
$$
SELECT DISTINCT cma.movie_id, cma.category_id
FROM categories_with_maximum_audience() cma
WHERE cma.category_id IN (SELECT fc.category_id
                          FROM up_to_n_user_favourite_categories(n, u_id) fc)
LIMIT n
$$;


SELECT *
FROM up_to_n_movie_recommendations_for_user(10, 123)
```

Листинг 1: recommendations.sql

4. Хранимая процедура 2

Задание: Вывести динамику 10 наиболее популярных категорий за последние три года: количество подписок, изменение относительно предыдущего года в абсолютном и относительном выражении, место в рейтинге и изменение места по сравнению с предыдущим годом.

В листинге 2 представлена процедура `top_10_categories_within_3_last_years()`, которая формирует выборку в соответствии с заданием.

```sql
CREATE OR REPLACE FUNCTION year_category_subscription_count(year_ago SMALLINT)
  RETURNS TABLE
          (
            year               SMALLINT,
            category_id        SMALLINT,
            subscription_count BIGINT,
            place              SMALLINT
```

```
8           )
9   LANGUAGE SQL
10 AS
11 $$
12 SELECT year_ago        AS year,
13        mc.category_id AS category_id,
14        count(s.id)    AS subscription_count,
15        (row_number() OVER (ORDER BY count(s.id) DESC))::SMALLINT
16 FROM movie_category mc
17        JOIN subscription_movie sm
18             ON mc.movie_id = sm.movie_id
19        JOIN subscription s
20             ON sm.subscription_id = s.id
21 WHERE floor(extract(EPOCH FROM age(now(), s.started)) / 3600 / 24 / 365) = year_ago
22 GROUP BY mc.category_id
23 ORDER BY subscription_count DESC
24 LIMIT 10
25 $$;
26
27
28 CREATE OR REPLACE FUNCTION top_10_categories_within_3_last_years()
29   RETURNS TABLE
30           (
31             category_id                    SMALLINT,
32             curr_place                     SMALLINT,
33             prev_place                     SMALLINT,
34             curr_prev_place_delta          SMALLINT,
35             curr_subscr_cnt                BIGINT,
36             prev_subscr_cnt                BIGINT,
37             curr_prev_subscr_cnt_delta     BIGINT,
38             curr_prev_subscr_cnt_rel       NUMERIC,
39             prevprev_place                 SMALLINT,
40             curr_prevprev_place_delta      SMALLINT,
41             prevprev_subscr_cnt            BIGINT,
42             curr_prevprev_subscr_cnt_delta BIGINT,
43             curr_prevprev_subscr_cnt_rel   NUMERIC
44           )
45   LANGUAGE SQL
46 AS
47 $$
48 SELECT curr.category_id                                              AS category_id,
49        curr.place                                                    AS curr_place,
50        prev.place                                                    AS prev_place,
51        (-curr.place + prev.place)                                    AS curr_prev_place_delta,
52        curr.subscription_count                                       AS curr_subscr_cnt,
53        prev.subscription_count                                       AS prev_subscr_cnt,
54        (curr.subscription_count - prev.subscription_count)           AS curr_prev_subscr_cnt_delta,
55        round((((curr.subscription_count -
56                 prev.subscription_count)::FLOAT)
57          / prev.subscription_count::FLOAT)::NUMERIC, 2)              AS curr_prev_subscr_cnt_rel,
58        prevprev.place                                                AS prevprev_place,
59        (-curr.place + prevprev.place)                                AS curr_prevprev_place_delta,
60        prevprev.subscription_count                                   AS prevprev_subscr_cnt,
61        (curr.subscription_count - prevprev.subscription_count) AS
      curr_prevprev_subscr_cnt_delta,
62        round((((curr.subscription_count -
63                 prevprev.subscription_count)::FLOAT)
64          / prevprev.subscription_count::FLOAT)::NUMERIC, 2)          AS curr_prevprev_subscr_cnt_rel
65
66 FROM year_category_subscription_count(0::SMALLINT) curr
67        LEFT JOIN year_category_subscription_count(1::SMALLINT) prev
68                  ON curr.category_id = prev.category_id
69        LEFT JOIN year_category_subscription_count(2::SMALLINT) prevprev
70                  ON curr.category_id = prevprev.category_id
71 ORDER BY curr_place
72 $$;
73
74 SELECT *
75 FROM top_10_categories_within_3_last_years()
```

Листинг 2: top-categories.sql

5. Выводы

В результате работы были изучены возможности языка PL/pgSQL, и были разработаны хранимые процедуры, формирующие выборки в соответствии с заданием преподавателя.