МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Интеллектуальные системы информационной безопасности»

Политики безопасности баз данных

Практическая работа № 3

ОТЧЁТ

Выполнил студент группы
БСБО-07-20
Любовский С.В.

Москва 2023

Выполнение задания 1.

1. Создать базу данных с именем vacuum_db.

   CREATE DATABASE vacuum_db;

2. Создать таблицу users, отключив параметр автоочистки (CREATE TABLE ... WITH (autovacuum_enabled = off);), со следующими полями:
   id: уникальный идентификатор пользователя (integer, primary key, auto-increment).
   username: имя пользователя (varchar(255)).
   email: электронный адрес пользователя (varchar(255)).
   category: категория (char(3))

   CREATE TABLE users (
   id serial primary key,
   username varchar(255),
   email varchar(255),
   category char(3)
) WITH (autovacuum_enabled = off);

3. Написать скрипт заполняющий таблицу users 1000000 рандомными записями, в поле category всегда должна находиться запись ‘FOO’.

   INSERT INTO users (username, email, category)
   SELECT gen_random_uuid() as username, gen_random_uuid() as email, 'FOO'::char(3) AS category
   FROM generate_series(1,1000000);

4. Используя оператор Explain выведите из таблицы users все записи, которые в поле category имеют значение ‘FOO’;

   EXPLAIN SELECT * FROM users WHERE category = 'FOO';

```
vacuum_db=# EXPLAIN SELECT * FROM users WHERE category = 'FOO';
                          QUERY PLAN
---------------------------------------------------------------
 Seq Scan on users  (cost=0.00..15536.02 rows=500 width=1052)
   Filter: (category = 'FOO'::bpchar)
(2 rows)
```

   Видно, что планировщик запросов сильно промахивается с оценкой количества строк.

5. Выполните команду ANALYZE;

ANALYZE users;

6. Используя оператор Explain выведите из таблицы users все записи, которые в поле category имеют значение ‘FOO’;

   EXPLAIN SELECT * FROM users WHERE category = 'FOO';

```
vacuum_db=# EXPLAIN SELECT * FROM users WHERE category = 'FOO';
                          QUERY PLAN
---------------------------------------------------------------
 Seq Scan on users  (cost=0.00..26786.00 rows=1000000 width=82)
   Filter: (category = 'FOO'::bpchar)
(2 rows)
```

   После актуализации статистики, планировщик правильно оценивает стоимость запроса и количество строк.

7. Отличаются ли методы доступа к данным и почему?

   Методы доступа к данным не отличаются, но планировщик грамотно оценивает стоимость запроса и количество строк в выборке.

8. Временно уменьшите значение maintenance_work_mem чтоб оно стало равно 1MB (не забудьте выполнить функцию pg_reload_conf())

   SET maintenance_work_mem to '1MB';
   SELECT pg_reload_conf();

9. Измените значение поля category на ‘BPP’

   UPDATE users SET category = 'BPP';

10. Запустите очистку VACUUM VERBOSE. Заодно через небольшое время в другом сеансе обратитесь к pg_stat_progress_vacuum.

    VACUUM VERBOSE;
    -- в другом сеансе
    SELECT * FROM pg_stat_progress_vacuum;

```
vacuum_db=# select * from pg_stat_progress_vacuum;
 pid | datid |  datname  | relid |     phase      | heap_blks_total | heap_blks_scanned | heap_blks_vacuumed | index_vacuum_count | max_dead_tuples |
 num_dead_tuples
-----+-------+-----------+-------+----------------+-----------------+-------------------+--------------------+--------------------+-----------------+-----------------
 312 | 24610 | vacuum_db | 24612 | vacuuming heap |           28572 |              7479 |               7360 |                  3 |          174761 |
         174510
(1 row)
```

11. Верните значение maintenance_work_mem к исходному значению.

    Стандартное значение – 64MB

SET maintenance_work_mem to '64MB';
SELECT pg_reload_conf();

Выполнение задания 2.

1. Узнать текущий размер файла данных таблицы users при помощи функции: pg_size_pretty(pg_table_size('название таблицы'))

   SELECT pg_size_pretty(pg_table_size('users'));

```
vacuum_db=# SELECT pg_size_pretty(pg_table_size('users'));
 pg_size_pretty
----------------
 223 MB
(1 row)
```

2. Удалите 90% случайных строк (Случайность важна, чтобы в каждой странице остались какие-нибудь не удаленные строки)

   Воспользуемся TEMPSAMLE и методом формирования выборки BERNOULLI для получения случайной выборки 90% таблицы.

   DELETE FROM users WHERE id IN (
   SELECT id FROM users TABLESAMPLE BERNOULLI (90)
   );

3. Выполните очистку

   VACUUM;

4. Ещё раз узнайте текущий размер файла данных таблицы users и сравните его с первым пунктом. Объясните результат.

   SELECT pg_size_pretty(pg_table_size('users'));

```
vacuum_db=# SELECT pg_size_pretty(pg_table_size('users'));
 pg_size_pretty
----------------
 223 MB
(1 row)
```

   Vacuum очищает страницы от старых данных оставляя пустые места и не сжимает страницы. Размер таблицы остается такой же.

5. Заново заполните таблицу и повторите пункты 1 и 2.
6. Выполните полную очистку

VACUUM FULL;

7. Ещё раз узнайте текущий размер файла данных таблицы users и сравните его с результатом пункта 5. Объясните результат.

   SELECT pg_size_pretty(pg_table_size('users'));

```
vacuum_db=# SELECT pg_size_pretty(pg_table_size('users'));
 pg_size_pretty
----------------
 12 MB
(1 row)
```

   Произошла очистка таблицы и сжатие страниц (удаление пустот). Размер страниц уменьшился.

Выполнение задания 3.

1. Включите параметр автоочистки в таблице users

   ALTER TABLE users SET (
   autovacuum_enabled = true
   );

2. Настройте автоочистку на запуск при изменении 10 % строк, время «сна» — одна секунда (autovacuum_vacuum_threshold = 0, autovacuum_vacuum_scale_factor = 0.1, autovacuum_naptime = '1s')

   Все эти настройки прописываются в файле postgresql.conf, после чего postgres необходимо перезагрузить.

3. Заполните таблицу users до 1000000 записей.

   INSERT INTO users (username, email, category)
   SELECT gen_random_uuid() as username, gen_random_uuid() as email, 'FOO'::char(3) AS category
   FROM generate_series(1,1000000);

4. Узнать текущий размер файла данных таблицы users при помощи функции: pg_size_pretty(pg_table_size('название таблицы'))

```
vacuum_db=# SELECT pg_size_pretty(pg_table_size('users'));
 pg_size_pretty
----------------
 112 MB
(1 row)
```

5. Напишите скрипт который двадцать раз с интервалом в несколько секунд изменяет по 5 % случайных строк. Каждое изменение выполняйте в отдельной транзакции.

```
DO
$BODY$
BEGIN
   FOR i IN 1..20 LOOP
      DELETE FROM users WHERE id IN (
         SELECT id FROM users TABLESAMPLE BERNOULLI (5)
      );
      COMMIT;
      PERFORM pg_sleep(2);
   END LOOP;
END
$BODY$
LANGUAGE plpgsql;
```

6. При помощи **pg_stat_all_tables** узнайте сколько раз выполнялась автоочистка (**autovacuum_count**)

Результат до применения скрипта

```
vacuum_db=# select schemaname,relname,last_autovacuum,autovacuum_count from pg_stat_all_tables where relname = 'users';
 schemaname | relname |        last_autovacuum        | autovacuum_count
------------+---------+-------------------------------+------------------
 public     | users   | 2023-03-20 16:51:34.030483+00 |                1
(1 row)
```

Результат после применения скрипта

```
vacuum_db=# select schemaname,relname,last_autovacuum,autovacuum_count from pg_stat_all_tables where relname = 'users';
 schemaname | relname |        last_autovacuum        | autovacuum_count
------------+---------+-------------------------------+------------------
 public     | users   | 2023-03-20 16:56:59.209561+00 |                2
(1 row)
```

7. Сравнить размеры таблицы до и после обновлений

```
vacuum_db=# SELECT pg_size_pretty(pg_table_size('users'));
 pg_size_pretty
----------------
 112 MB
(1 row)
```

Размер таблицы не изменился, так как не был выполнен глубокий вакуум.

8. Совпадают ли результаты с ожидаемыми и как их объяснить?

В связи с установлением параметра autovacuum_vacuum_threshold количество автоочисток уменьшилось, что сложилось с фактором autovacuum_vacuum_scale_factor порог срабатывания очистки срабатывает в 10%, при том, что мы изменяли лишь 5%.