Cheatsheets / **Advanced PostgreSQL**

codecademy

Database Maintenence

Updates and Deletes Effect on Table Size

When using PostgreSQL, the size of database tables can grow unexpectedly large with routine UPDATE and DELETE operations.

```
-- Step 1. Generate A New Table
CREATE TABLE rand as (
    SELECT id, random() as score
    FROM generate_series(1, 100000) as id
);

-- Step 2. Check Table Size
SELECT pg_size_pretty(
  pg_total_relation_size('rand')
) as table_size;

/*
+------------+
| table_size |
+------------+
| 4360 kB    |
+------------+
*/

-- Step 3. Update Tuples (~20% of values)
UPDATE rand SET score = 1 where score >
.8;
/* UPDATE 19925 */

-- Step 4. Check Table Size w. ~20% of
tuples updated
SELECT pg_size_pretty(
  pg_total_relation_size('rand')
) as table_size;

/*
+------------+
| table_size |
+------------+
| 6080 kB    |
+------------+
*/
```

PostgreSQL Dead Tuples

In PostgreSQL, when a row is deleted or updated, PostgreSQL creates so-called Dead tuples. Dead tuples are not referenced in the current version of our databases' tables, but still occupy space on disk.

```
--  Dead tuples contribute to the size of a table but aren't displayed to the DB user: You can check the number of dead tuples with the internal PostgreSQL statistic tables.

SELECT
  schemaname,
  relname,
  n_dead_tup
FROM pg_catalog.pg_stat_all_tables
WHERE relname = 'rand';

/*
+------------+---------+------------+
| schemaname | relname | n_dead_tup |
+------------+---------+------------+
| public     | rand    |      10000 |
+------------+---------+------------+
*/
```

PostgreSQL Vacuuming

In PostgreSQL, to reclaim space from dead tuples, you can use `VACCUUM`, `VACCUM ANALYZE`, or `VACCUM FULL`, each comes with a different strategy for clearing dead tuples.

Importance of VACUUM

In PostgreSQL, It's important to occasionally `VACUUM` tables to keep database queries performant and use database space efficiently.

```
-- Depending on the status of a table's
inserts, deletes, and updates, a `VACUUM`
can reduce space used on disk, or it may
just clear space in the same table for
new inserts.

-- Before VACUUM
SELECT
  schemaname,
  relname,
  n_dead_tup
FROM pg_catalog.pg_stat_all_tables

/*
+------------+---------+------------+
| schemaname | relname | n_dead_tup |
+------------+---------+------------+
| public     | rand    |      10000 |
+------------+---------+------------+
*/


VACUUM mocked_data.time_series;


-- Same Query -> After VACUUM
/*
+------------+---------+------------+
| schemaname | relname | n_dead_tup |
+------------+---------+------------+
| public     | rand    |          0 |
+------------+---------+------------+
*/
```

PostgreSQL Analyze

In PostgreSQL, `ANALYZE` collects statistics about the contents of tables in the database, and stores the results in the system catalog so PostgreSQL can determine the efficient way to execute a query.

```
-- The statement to analyze a table named `schema.table`:
ANALYZE schema.table;
```

VACUUM in PostgreSQL

In PostgreSQL, plain `VACUUM` can run in parallel with database operations, but `VACUUM` does not always fully reduce table sizes. Instead, it marks the space on disk as safe to overwrite with new data.

```
-- VACUUM `schemaname.tablename` with the below:
VACUUM schemaname.tablename;
```

VACUUM FULL in PostgreSQL

In PostgreSQL, `VACUUM FULL` should be used to fully reclaim database space. However, `VACUUM FULL` rewrites the entire contents of the table into a new location on disk with no extra space allocated. This is an expensive operation and should be used sparingly.

```
-- Step 1. Check Status of Table - 44.2K
Dead Tuples

SELECT
  schemaname,
  relname,
  n_dead_tup
FROM pg_catalog.pg_stat_all_tables
WHERE relname = 'rand';

/*
+------------+---------+------------+
| schemaname | relname | n_dead_tup |
+------------+---------+------------+
| public     | rand    |      44157 |
+------------+---------+------------+
*/

-- Step 2. Run `VACUUM FULL`
VACUUM FULL rand;

-- Step 3. Confirm dead tuples removed.
SELECT
  schemaname,
  relname,
  n_dead_tup
FROM pg_catalog.pg_stat_all_tables
WHERE relname = 'rand';

/*
+------------+---------+------------+
| schemaname | relname | n_dead_tup |
+------------+---------+------------+
| public     | rand    |          0 |
+------------+---------+------------+
*/
```

Vacuum and Autovacuum in PostgreSQL

PostgreSQL has a feature called autovacuum, which automatically runs `VACUUM` and `ANALYZE` commands. When enabled, autovacuum checks for tables that have had a large number of inserted, updated or deleted tuples.

```
-- Autovacuum is enabled on most Database
Instances, consider the following:
-- Step 1. Table shows no Autovacuum time
select
        schemaname,
        relname,
  last_autovacuum
FROM pg_catalog.pg_stat_all_tables
WHERE relname = 'rand';

/*
+------------+---------+-----------------
--+
| schemaname | relname |  last_autovacuum
|
+------------+---------+-----------------
--+
| public     | rand    |
|
+------------+---------+-----------------
--+
*/

-- Step 2. Duplicate all rows of a table
(`rand`) by re-inserting all rows...
INSERT INTO rand (
  SELECT * FROM rand
);

-- Step 3. This is a large insert,
checking autovacuum time
select
        schemaname,
        relname,
  last_autovacuum
FROM pg_catalog.pg_stat_all_tables
WHERE relname = 'rand';

/*
```

```
+------------+---------+-----------------
--------------+
| schemaname | relname |
last_autovacuum        |
+------------+---------+-----------------
--------------+
| public     | rand    | 2021-07-11
06:24:11.957292-04 |
+------------+---------+-----------------
--------------+
*/
```

PostgreSQL Truncate

In PostgreSQL, to improve performance of large deletes, `TRUNCATE` is preferable to `DELETE`, `TRUNCATE` is faster and automatically reclaims the space on disk.

```
-- Step 1. Generate Table
CREATE TABLE rand as (
    SELECT id, random() as score
    FROM generate_series(1, 100000) as id
);

-- Step 2. Check Table Size
select pg_size_pretty(
  pg_total_relation_size('rand')
) as table_size;

/*
+------------+
| table_size |
+------------+
| 4360 kB    |
+------------+
*/

-- Step 3. Option 1: Remove all contents
from table, reclaims no space
DELETE FROM rand WHERE TRUE;

select pg_size_pretty(
  pg_total_relation_size('rand')
) as table_size;
/*
+------------+
| table_size |
+------------+
| 4360 kB    |
+------------+
*/


-- Step 3. Option 2: With Truncate -
Faster and Automatically Reclaims Space
TRUNCATE rand;
```

```
select pg_size_pretty(
  pg_total_relation_size('rand')
) as table_size;

/*
+------------+
| table_size |
+------------+
| 0 bytes    |
+------------+
*/
```

PostgreSQL All Table Statistics

In PostgreSQL, you can monitor table statistics by querying the view `pg_stat_all_tables`. This view contains statistics like number of dead and live tuples, number of rows inserted, and last vacuum or autovacuum time.

```
-- Sample query to get table statistics
from `pg_stat_all_tables`
SELECT
        schemaname,
        relname,
  n_live_tup,
  n_tup_upd,
  n_tup_del,
  last_vacuum,
  last_autovacuum
FROM pg_catalog.pg_stat_all_tables
WHERE relname = 'clicks';

/*
+------------+---------+------------+----
-------+-----------+--------------------
----------+-----------------+
| schemaname | relname | n_live_tup |
n_tup_upd | n_tup_del |
last_vacuum          | last_autovacuum |
+------------+---------+------------+----
-------+-----------+--------------------
----------+-----------------+
| public     | clicks  |       8200 |
0 |         0 | 2021-06-11
00:23:21.187128-04 |                 |
+------------+---------+------------+----
-------+-----------+--------------------
----------+-----------------+
*/
```

Print Share