Information Schema and pg_catalog

1. Information Schame

Information schema consists of a set of **views** that contain information about the objects defined in the current database. The information schema is defined in the SQL standard and can therefore be expected to be portable and remain stable.

Information schema contains the metadata of a database. It allows users to gain knowledge of the structure of a database before comming up with queries and other manipulations on the tables.

Note that different DBMS products may have different implementations.
Here we introduce some important views in information schema using PostgreSQL.

- Tables

  The view tables contains all tables and views defined in the current database. You can use information_schema.views to see only views.

  E.g. To obtain all table and views in information schema:

  ```
  select * from information_schema.tables where table_schema =
  'information_schema';
  ```

- Columns

  The view columns contains information about all table columns (or view columns) in the database. System columns (oid, etc.) are not included.

  E.g. To obtain all columns in table movies :

  ```
   select *
  from information_schema.columns where table_name = 'movies';
  ```

- Schemata

  The view schemata contains all schemas in the current database that the current user has access to.

  ```
   select *
  from information_schema.schemata;
  ```

- Routine

  The view routines contains all functions and procedures in the current database.

  ```
   select *
  from information_schema.routines;
  ```

- Trigger

  The view triggers contains all triggers defined in the current database on tables and views that the current user owns or has some privilege other than SELECT on.

  ```
   select *
  from information_schema.triggers;
  ```

2. Catalog & System Administration Functions (PostgreSQL)

Tables in pg_calalog and the system administration fuctions provide powerful statistical tools to analyze our database. They helps us design good indexes and efficient queries.

pg_catalog

To see all the tables and views in pg_catalog :

```
 select *
from information_schema.tables where table_schema = 'pg_catalog';
```

Examples:

- pg_stats pg_stats provide statistical information of columns. It's also used by the *query optimizer*.

  ```
   select * from pg_stats where schemaname = 'public' and tablename =
  'movies';
  ```

- pg_statio_user_tables

  shows I/O statistics of each table in the current database.

  ```
   select * from pg_statio_user_tables where schemaname = 'public';
  ```

- pg_index
- pg_class
- pg_attribute

System Administration Functions

Please refer to https://www.postgresql.org/docs/9.4/functions-admin.html for detailed descriptions. Examples:

- pg_database_size(oid) bigint
- pg_database_size(name) bigint

  ```
  select oid from pg_database where datname = 'cs307';
  select pg_database_size(16386);
  select pg_database_size('cs307');
  ```

- pg_tablespace_size(oid) bigint
- pg_tablespace_size(name) bigint

```
select pg_tablespace_size('pg_default')/1024/1024 as "size in M";
select pg_size_pretty(pg_tablespace_size('pg_default'));
```

- pg_table_size(regclass) bigint
- pg_indexes_size(regclass) bigint All indexes
- pg_total_relation_size(regclass)

```
select pg_size_pretty(pg_table_size('movies'));
select pg_size_pretty(pg_indexes_size('movies'));
select pg_size_pretty(pg_total_relation_size('movies'));
```

- pg_column_size(any)

```
select pg_column_size('hello world');
```

Exercise

You are asked to create a view called DATABASE_STORAGE that will return the following information for the current database (only include the tables in your own schema) and the result will be in descending order according to the total size:

SCHEMA_NAME (you will write with a format as "schema_name"."table_name")

DATA_SIZE (you will use function pg_size_pretty() to return the information)

INDEX_SIZE (using pg_size_pretty())

TOTAL_SIZE (using pg_size_pretty())

PERCENT (percentage of the total, rounded to 1 decimal)

Please use exactly these view and column names ...

Sample Answer:

```
create or replace view DATABASE_STORAGE as
select table_schema || '.' || table_name,
       pg_size_pretty(pg_table_size(cast(table_name as varchar))) table_size,
       pg_size_pretty(pg_indexes_size(cast(table_name as varchar))) index_size,
       pg_size_pretty(pg_total_relation_size(cast(table_name as varchar)))
total_relation_size,
       round(pg_total_relation_size(cast(table_name as varchar)) /
sum(pg_total_relation_size(cast(table_name as varchar))) over () * 100, 1)
from information_schema.tables
where table_schema = current_schema
  and table_type = 'BASE TABLE'
order by pg_total_relation_size(cast(table_name as varchar)) desc;
```

Reference

This tutorial is based on tutorial of Lab 11 CS307 Spring 2018 by Yueming Zhu and updated by Shijie Chen.

- https://www.postgresql.org/docs/current/information-schema.html
- https://www.postgresql.org/docs/9.4/functions-admin.html