KOREA UNIVERSITY
DATAX LAB

Chapter 16 - Lab

Query Optimization 2

Preliminaries

- Statistics for query planning and optimization
  - System relations for storing statistics: pg_class, pg_stats, …
- PostgreSQL stores various statistics
  - Representative useful statistics
    - Most common values/frequencies and histogram
  - Most common values and most common frequencies
    - A list of the most common values in the column and their frequencies
      - The frequency of a value is the number of its occurrences divided by total numbers of rows

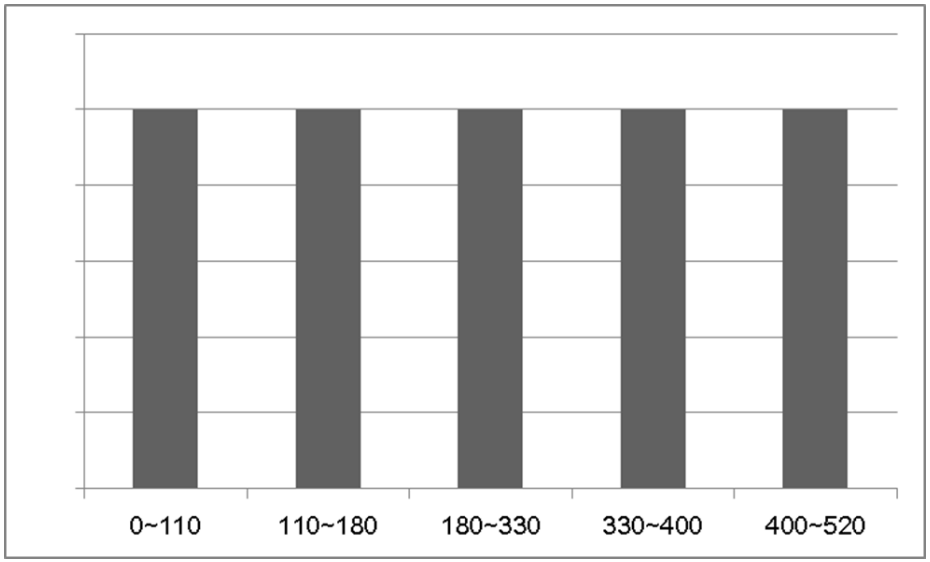
Most common values	Most common frequencies
580	0.004
230	0.003
410	0.002
350	0.002
90	0.002

<Most common values/frequencies example >

Preliminaries

PostgreSQL stores various statistics

- Histogram
  - A list of values the divide the column's values into groups of approximately equal population
  - The value in most common values are omitted



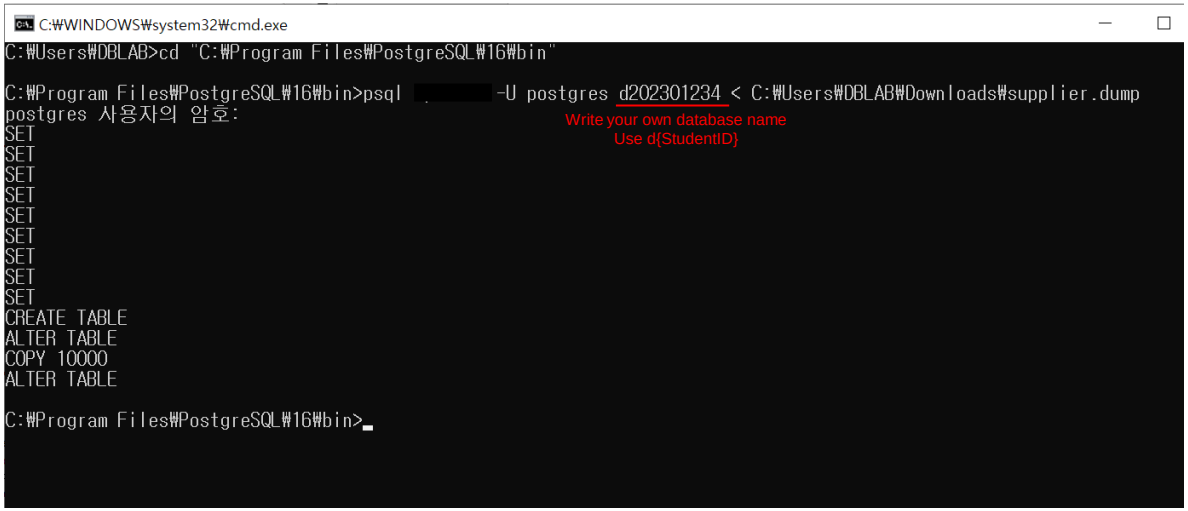
<Histogram example >

Lab Setup

- Remove table “supplier” from PostgreSQL
  - DROP TABLE supplier;
- Download the file from blackboard
  - “supplier.dump”

Lab Setup (Windows)

- Open Command Prompt (cmd.exe) and type the following commands:
  1. cd C:\Program Files\PostgreSQL\16\bin
     - This is the default path. If you installed it somewhere else, go to that path.
  2. psql -U postgres postgres < [filepath]\supplier.dump
     - For [filepath], type the path where you downloaded “supplier.dump”.
  3. Type your own PostgreSQL password



Lab Setup (Mac OS X)

- Open Terminal and type the following commands:
  1. cd /Library/PostgreSQL/16/bin
     - This is the default path. If you installed it somewhere else, go to that path.
  2. ./psql -U postgres postgres < [filepath]/supplier.dump
     - For [filepath], type the path where you downloaded “supplier.dump”.
  3. Type your own PostgreSQL password

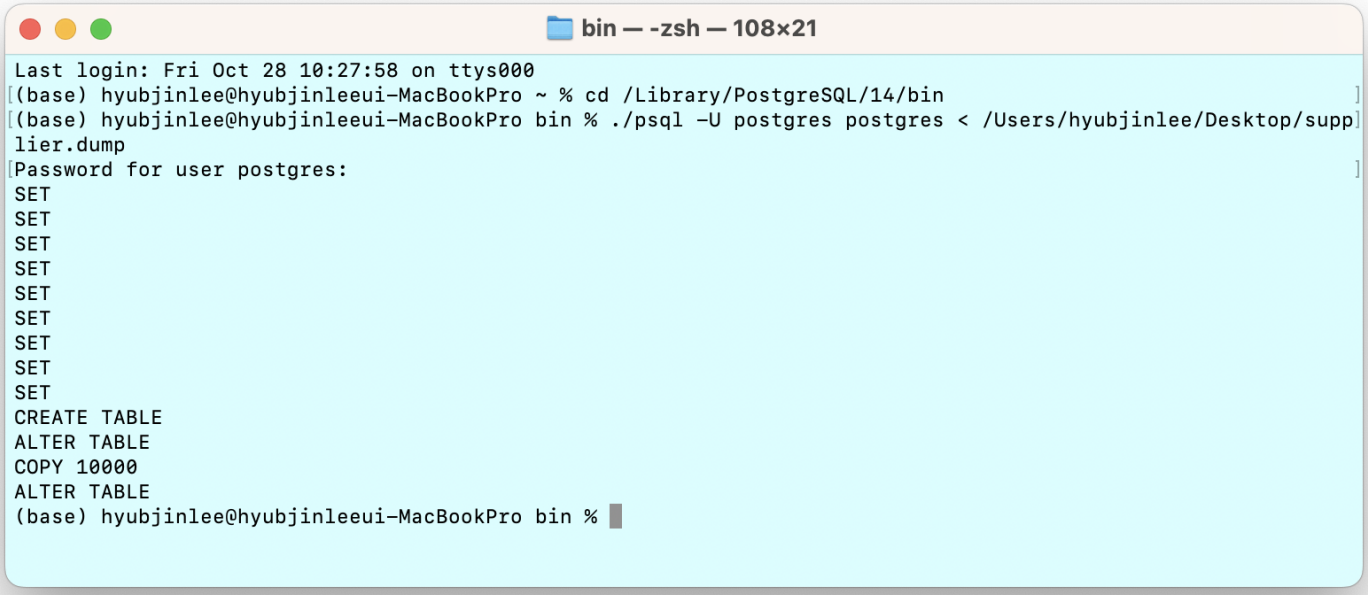


Table Information

- “supplier” has 10,000 rows
- “supplier” ’s schema is as follows:

Attribute	Data Type	Cardinality	Features
s_suppkey	integer	10,000	primary key
s_name	character varying(200)	10,000	
s_address	character varying(200)	10,000	
s_nationkey	integer	25	
s_phone	character varying(200)	10,000	
s_acctbal	double precision	9,955	
s_comment	character varying(200)	10,000	

Lab Setup

- Execute PostgreSQL SQL Shell (psql) and login your database
  - Server [localhost]: Press the enter key
  - Database [postgres]: Press the enter key
  - Port [5432]: Press the enter key
  - Username [postgres]: Press the enter key
  - Password for user postgres: Type your own password
- Type on psql command line
  - SET enable_bitmapscan=false;
  - SET max_parallel_workers_per_gather=0;

Exercise 1

- Estimate the results of the following queries and compare it to the actual results (Be sure to show the calculation process in your homework)
  a. SELECT count(*) FROM supplier WHERE s_suppkey<=350;
  b. SELECT count(*) FROM supplier WHERE s_acctbal<=405.68;

  - Hint
    - Note that '*s_suppkey'* is the primary key of the supplier relation
    - Getting the number of tuples of the supplier relation
      - SELECT reltuples FROM pg_class WHERE relname='supplier';
    - Getting the histogram of a column of the supplier relation (num_bucket =100)
      - SELECT histogram_bounds FROM pg_stats WHERE tablename='supplier' AND attname='[column name]';
    - Getting the most common values and most common frequencies of a column of the supplier relation
      - SELECT most_common_vals, most_common_freqs FROM pg_stats WHERE tablename='supplier' AND attname='[column name]'

Exercise 2

- What is a good way to evaluate the following queries
  a. SELECT * FROM supplier WHERE s_suppkey<=350;
  b. SELECT * FROM supplier WHERE s_suppkey>350;

- Verify whether your estimation is correct by using ‘EXPLAIN ANALYZE’

Lab Setup

- Create four synthetic data tables that has (100 / 1,000 / 10,000 / 100,000) rows with values between 0 and 100
  - CREATE TABLE t1(val integer);
  - CREATE TABLE t2(val integer);
  - CREATE TABLE t3(val integer);
  - CREATE TABLE t4(val integer);

  - INSERT INTO t1(val) SELECT random()*100 FROM (SELECT generate_series(1,100)) as T;
  - INSERT INTO t2(val) SELECT random()*100 FROM (SELECT generate_series(1,1000)) as T;
  - INSERT INTO t3(val) SELECT random()*100 FROM (SELECT generate_series(1,10000)) as T;
  - INSERT INTO t4(val) SELECT random()*100 FROM (SELECT generate_series(1,100000)) as T;

Exercise 3

- Estimate how PostgreSQL will determine the join order for the next two queries
  a. SELECT count(*) FROM t1 NATURAL JOIN t2 NATURAL JOIN t3 NATURAL JOIN t4;
  b. SELECT count(*) FROM t4 NATURAL JOIN t3 NATURAL JOIN t2 NATURAL JOIN t1;

- Verify whether your estimation is correct by using ‘EXPLAIN ANALYZE’

Exercise 4

- Force a join order by using the command below
  - SET join_collapse_limit=1;

- Estimate which of the two queries will be executed faster
  a. SELECT count(*) FROM t1 NATURAL JOIN t2 NATURAL JOIN t3 NATURAL JOIN t4;
  b. SELECT count(*) FROM t4 NATURAL JOIN t2 NATURAL JOIN t3 NATURAL JOIN t1;

- Verify whether your estimation is correct by using ‘EXPLAIN ANALYZE’

Homework

- Complete today’s practice exercises
- Write your queries and take screenshots of execution results
- Submit your report on blackboard
  - 10:29:59, November 19th, 2024
  - **Only PDF files** are accepted
  - **No late submission**

End of Lab