**Query Performance Profiling**

Yuping Yang

Contents

[Problem Statement 2](#_Toc166258898)

[Description of the preliminary experiments 2](#_Toc166258899)

[Query 1 2](#_Toc166258900)

[Query 2 2](#_Toc166258901)

[Query 3 3](#_Toc166258902)

[Explanations and Reasoning of the observed Phenomena 4](#_Toc166258903)

[Experiments Design Explanation 4](#_Toc166258904)

[Experiments Design Factors 5](#_Toc166258905)

[Query Results Format 5](#_Toc166258906)

[ANOVA Analysis 6](#_Toc166258907)

[Appendix 7](#_Toc166258908)

[Platform 7](#_Toc166258909)

[Data tables 7](#_Toc166258910)

[Query 1 7](#_Toc166258911)

[Query 2 8](#_Toc166258912)

[Query 3 8](#_Toc166258913)

[Block 1(query 1, 2, 3, in that order) 8](#_Toc166258914)

[Block 2 (query 2, 3, 1, in that order) 11](#_Toc166258915)

[Block 3 (query 3, 1, 2, in that order) 14](#_Toc166258916)

# Problem Statement

The experiment design problem is to test and compare the query performance for 3 similar queries in a PostgreSQL database.

# Description of the preliminary experiments

Try to test and compare the execution time of 3 queries, Query 1, Query 2, Query 3, from the same two tables in a PostgreSQL database. The experiments are done on a PC with an 2.6GHz i7 processor and Windows 11 system. The database server is a PostgreSQL server 16.2-1. There are two tables used in the queries. Table acc\_accession has 68,942,078 rows and 13 columns. Table acc\_accessionreference has 9116737 rows and 6 columns. Both acc\_accession and acc\_accessionreference can be joined through \_accession\_key field.

The 3 queries are:

## Query 1

with q1 as (

select \_accession\_key

from mgd.acc\_accessionreference)

select count(\*)

into r1

from mgd.acc\_accession acc

join q1

on acc.\_accession\_key = q1.\_accession\_key;

## Query 2

with q2 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key

from mgd.acc\_accessionreference)

select count(\*)

into r2

from mgd.acc\_accession acc

join q2

on acc.\_accession\_key = q2.\_accession\_key;

## Query 3

with q3 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key,

creation\_date, modification\_date

from mgd.acc\_accessionreference)

select count(\*)

into r3

from mgd.acc\_accession acc

join q3

on acc.\_accession\_key = q3.\_accession\_key;

Through some preliminary experiments (query executions), I found that

1. If a query is executed for the first time, or executed after a long time, for example, after several hours of not using the PC, the execution time is long. But if I repeat the execution immediately after the execution of a query, the execution time can have a up to 50% time reduction, that is, in this situation, execution time may cut to half in extreme case.
2. The execution time reduction is also observed even the first and second query are different but similar to each other.
3. In general, the longer the time gap between two execution times, the less the execution time reduction in the second execution is observed.
4. In general, the more activities on the PC between two executions, the less the execution time reduction in the second execution is observed. The other activities, including, but not limited to web browsing between the two executions to be compared.

# Explanations and Reasoning of the observed Phenomena

PostgreSQL server has a shared buffer that can cache the query result. It’s minimal recommended size is 124MB. Complete elimination of the shared buffer is not recommended.

This may explain the observed phenomenon (1) above. But not fully explained the observed phenomenon (2) above.

PostgreSQL server appears to share some system resources with other programs unrelated to PostgreSQL in the same Windows PC. If this is true, and I believe it is true, explains (2), (3), (4) above.

# Experiments Design Explanation

I made 3 SQL blocks, each containing all 3 queries. A SQL block can be executed as one program.

SQL block 1 contains query 1, query 2, query 3, in that order

SQL block 2 contains query 2, query 3, query 1, in that order

SQL block 3 contains query 3, query 1, query 2, in that order

The design is to balance the effect that for an execution of a block, the first query tends to take a little more time, and the second query tends to take less, and the third query tends to take even less time. This design gives each query an even chance of being the first query to be executed in a block.

Also, in each block, between queries, I placed statement:

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

This is to clean the shared buffer and any cached data, so that the subsequent queries in the same block can execute without or at least reduce the benefit of cached result.

Each block contains all 3 queries.

I ran block 1, block 2, block 3, in that order, and then ran block 1, block 2, block 3 in that order again. In this way, I reduced potential bias caused by the order of query execution when run different queries. Total, I ran 8 times. 8x3=24, total 24 experiments. The results are 24 numbers, representing 36 query execution times from 3 similar queries.

To consider the factor for “query execution one by one in succession” and “query execution after long paus of no query execution or after other computer use activities”. I repeat the above experiment 24 times. All settings of the experiments are the same, except between each execution (execution of a SQL block, which contains 3 queries), a time gap of at least 5 minutes is observed and a memory intensive program is executed to make sure that the memory used by PostgreSQL is squeezed to the minimum.

# Experiments Design Factors

By this experiment design, there are 3 factors:

Factor 1 (2 levels): Whether there is a long time gap (>5 min) and a memory intensive program run between experiments.

Factor 2 (3 levels): query execution orders, query1, query2, query3, i.e., (1, 2, 3) and then (2, 3, 1) and then (3, 1, 2), in that order. The order is enforced in a SQL block and I programmed 3 SQL blocks to enforce the 3 orders.

Factor 3 (3 levels): query 1, query 2, query 3

Response variable is the execution time collected during experiments.

# Query Results Format

Data file is profiling.txt

It is concatenated from profiling1.txt and profiloing2.txt

Profiling1.txt is factor1=1 (short succession runs)

Profiling2.txt is factor1=2 (at least 5 min gap and run other memory hungry programs in between)

In each of profiling1.txt and profiling2.txt, there are 3 groups of rows.

In each row group, factor2 is fixed, factor2 goes from 1 to 3

Each rows group has 24 rows.

These 24 rows are divided into 8 subgroups, each having 3 rows.

In each row subgroup, there are 3 rows, representing (i, j,1) (i,j,2) (i,j,3)

Where factor1=i and factor2=j.

The query results are collected into a text file with one column of values.

The data rows are indexed by index (i, j, k), where i in (1, 2), j in (1,2,3), k in (1,2,3)

Index component i represents factor 1 (long time gap or not)

Index component j represents factor 2 (query execute order 1, 2, 3)

Index component k represents factor 3 (query 1, or query 2, or query 3)

The index order is as such:

First row has index (1, 1, 1). That is, i=1, j=1, k=1. Then, increment of index start from right to the left. First increment k, up to 3. Then increment j, up to 3. And then increment I, up to 2.

And for each index (I,j,k), there are 8 rows.

For example, the first 3 rows of the data has identical index (1, 1, 1) (1, 1, 2) (1, 1, 3) and the first 24 rows repeat the index as of the first 3 rows. Total 24 rows with 3 different indices.

Then, the next 24 rows repeat the index from the first 24 rows, except the second index, corresponding to factor 2, is incremented to 2.

Total, there are 144 rows of values in the dataset as 2 x 3 x 3 x 8 = 144

# ANOVA Analysis

A screenshot of a computer

Description automatically generated

From the 3-way ANOVA analysis, it can be seen that factor has a very small p value while all other factors and cross factors have relatively large p value. From this analysis, I concluded that:

1. No significant difference between execution time of different queries from the same table
2. No significant difference between the order of execution of difference queries in a sequence of executions from the same table
3. There is significant difference between execution time of query execution with other activities run before query execution, or there is at least 5 minute of time gap before query execution.

Overall, the biggest influence to query performance is the influence from outside the database server.

# Appendix

## Platform

A PostgreSQL database server is installed on a PC with

16.2-1 Windows-x64) PostgreSQL server

Windows 11 Home operating system,

Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.60 GHz Processor

16GB RAM

-----------------

## Data tables

acc\_accession has 68,942,078 rows and 13 columns

acc\_accessionreference has 9116737 rows and 6 columns

acc\_accession and acc\_accessionreference can be joined through \_accession\_key field

-----------------

## Query 1

with q1 as (

select \_accession\_key

from mgd.acc\_accessionreference)

select count(\*)

into r1

from mgd.acc\_accession acc

join q1

on acc.\_accession\_key = q1.\_accession\_key;

----------

## Query 2

with q2 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key

from mgd.acc\_accessionreference)

select count(\*)

into r2

from mgd.acc\_accession acc

join q2

on acc.\_accession\_key = q2.\_accession\_key;

-----------

## Query 3

with q3 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key,

creation\_date, modification\_date

from mgd.acc\_accessionreference)

select count(\*)

into r3

from mgd.acc\_accession acc

join q3

on acc.\_accession\_key = q3.\_accession\_key;

-------------

## Block 1(query 1, 2, 3, in that order)

DO $$

DECLARE

start\_time TIMESTAMP;

end\_time TIMESTAMP;

duration1 INTERVAL;

duration2 INTERVAL;

duration3 INTERVAL;

r1 int;

r2 int;

r3 int;

BEGIN

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q1 as (

select \_accession\_key

from mgd.acc\_accessionreference)

select count(\*)

into r1

from mgd.acc\_accession acc

join q1

on acc.\_accession\_key = q1.\_accession\_key;

end\_time := clock\_timestamp();

duration1 := end\_time - start\_time;

----------------------------------------------------------

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q2 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key

from mgd.acc\_accessionreference)

select count(\*)

into r2

from mgd.acc\_accession acc

join q2

on acc.\_accession\_key = q2.\_accession\_key;

end\_time := clock\_timestamp();

duration2 := end\_time - start\_time;

----------------------------------------------------------

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q3 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key,

creation\_date, modification\_date

from mgd.acc\_accessionreference)

select count(\*)

into r3

from mgd.acc\_accession acc

join q3

on acc.\_accession\_key = q3.\_accession\_key;

end\_time := clock\_timestamp();

duration3 := end\_time - start\_time;

----------------------------------------------------------

RAISE NOTICE 'Query result % time %', r1, duration1;

RAISE NOTICE 'Query result % time %', r2, duration2;

RAISE NOTICE 'Query result % time %', r3, duration3;

END $$;

-------------

## Block 2 (query 2, 3, 1, in that order)

DO $$

DECLARE

start\_time TIMESTAMP;

end\_time TIMESTAMP;

duration1 INTERVAL;

duration2 INTERVAL;

duration3 INTERVAL;

r1 int;

r2 int;

r3 int;

BEGIN

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q2 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key

from mgd.acc\_accessionreference)

select count(\*)

into r2

from mgd.acc\_accession acc

join q2

on acc.\_accession\_key = q2.\_accession\_key;

end\_time := clock\_timestamp();

duration2 := end\_time - start\_time;

----------------------------------------------------------

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q3 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key,

creation\_date, modification\_date

from mgd.acc\_accessionreference)

select count(\*)

into r3

from mgd.acc\_accession acc

join q3

on acc.\_accession\_key = q3.\_accession\_key;

end\_time := clock\_timestamp();

duration3 := end\_time - start\_time;

----------------------------------------------------------

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q1 as (

select \_accession\_key

from mgd.acc\_accessionreference)

select count(\*)

into r1

from mgd.acc\_accession acc

join q1

on acc.\_accession\_key = q1.\_accession\_key;

end\_time := clock\_timestamp();

duration1 := end\_time - start\_time;

----------------------------------------------------------

RAISE NOTICE 'Query result % time %', r1, duration1;

RAISE NOTICE 'Query result % time %', r2, duration2;

RAISE NOTICE 'Query result % time %', r3, duration3;

END $$;

-------------

## Block 3 (query 3, 1, 2, in that order)

DO $$

DECLARE

start\_time TIMESTAMP;

end\_time TIMESTAMP;

duration1 INTERVAL;

duration2 INTERVAL;

duration3 INTERVAL;

r1 int;

r2 int;

r3 int;

BEGIN

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q3 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key,

creation\_date, modification\_date

from mgd.acc\_accessionreference)

select count(\*)

into r3

from mgd.acc\_accession acc

join q3

on acc.\_accession\_key = q3.\_accession\_key;

end\_time := clock\_timestamp();

duration3 := end\_time - start\_time;

----------------------------------------------------------

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q1 as (

select \_accession\_key

from mgd.acc\_accessionreference)

select count(\*)

into r1

from mgd.acc\_accession acc

join q1

on acc.\_accession\_key = q1.\_accession\_key;

end\_time := clock\_timestamp();

duration1 := end\_time - start\_time;

----------------------------------------------------------

PERFORM pg\_advisory\_unlock\_all();

RESET ALL;

start\_time := clock\_timestamp();

with q2 as (

select \_accession\_key, \_refs\_key, \_createdby\_key, \_modifiedby\_key

from mgd.acc\_accessionreference)

select count(\*)

into r2

from mgd.acc\_accession acc

join q2

on acc.\_accession\_key = q2.\_accession\_key;

end\_time := clock\_timestamp();

duration2 := end\_time - start\_time;

----------------------------------------------------------

RAISE NOTICE 'Query result % time %', r1, duration1;

RAISE NOTICE 'Query result % time %', r2, duration2;

RAISE NOTICE 'Query result % time %', r3, duration3;

END $$;