**Query Performance Profiling**

Yuping Yang

Contents

[Problem Statement 1](#_Toc166160823)

[Description of the preliminary experiments 2](#_Toc166160824)

[Explanations and Reasoning of the observed Phenomena 2](#_Toc166160825)

[Experiments Design 2](#_Toc166160826)

[Query results 3](#_Toc166160827)

[ANOVA Analysis 5](#_Toc166160828)

[Appendix 8](#_Toc166160829)

[Platform 8](#_Toc166160830)

[Data tables 8](#_Toc166160831)

[Query 1 8](#_Toc166160832)

[Query 2 8](#_Toc166160833)

[Query 3 9](#_Toc166160834)

[Block 1(query 1, 2, 3, in that order) 9](#_Toc166160835)

[Block 2 (query 2, 3, 1, in that order) 12](#_Toc166160836)

[Block 3 (query 3, 1, 2, in that order) 15](#_Toc166160837)

# 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.

Through some preliminary testing (experiments), I found that

1. If a query is executed for the first time, or executed after a long time, for example, after several hours, 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

I made 3 SQL blocks, each containing all 3 queries.

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 12 times. 12x3=36, total 36 experiments. The results are 36 numbers, representing 36 query execution times from 3 similar queries.

# Query results

The following are the results:

|  |  |
| --- | --- |
| row | query1,query2,query3 |
| 1 | 32.196836,26.799917,23.673738 |
| 2 | 25.09076,24.615621,27.423646 |
| 3 | 26.221316,23.35775,25.776975 |
| 4 | 22.08887,20.24331,20.137333 |
| 5 | 26.285219,24.389918,31.297548 |
| 6 | 30.39593,27.45738,25.776975 |
| 7 | 32.450399,38.504536,34.886932 |
| 8 | 29.796807,34.752938,31.445298 |
| 9 | 27.363795,28.369637,36.905693 |
| 10 | 30.053821,23.152757,22.141072 |
| 11 | 23.538129,22.890658,24.628442 |
| 12 | 24.680871,27.574666,26.295552 |
| 13 | 26.309098,23.108203,21.954799 |
| 14 | 24.676147,22.538573,31.026011 |
| 15 | 24.974906,36.996083,27.31156 |
| 16 | 30.237209,23.098815,22.104597 |
| 17 | 20.522875,20.83456,22.470693 |
| 18 | 23.204331,22.826447,21.98902 |
| 19 | 23.078849,21.884726,20.635991 |
| 20 | 22.549952,25.571441,24.388539 |
| 21 | 21.24216,25.373617,22.045887 |
| 22 | 23.580082,20.459172,20.19297 |
| 23 | 21.588172,27.306825,24.785626 |
| 24 | 25.319065,28.127265,26.662475 |
| 25 | 26.362981,24.98433,25.040282 |
| 26 | 20.06108,19.988207,22.448192 |
| 27 | 22.853495,25.23151,22.190933 |
| 28 | 27.221567,22.680488,23.395103 |
| 29 | 20.333009,20.25004,22.459006 |
| 30 | 22.160144,22.726759,21.363952 |
| 31 | 25.141441,27.705986,24.115168 |
| 32 | 25.321974,24.653369,30.587972 |
| 33 | 23.893349,30.344261,24.00536 |
| 34 | 31.399647,33.318846,33.302491 |
| 35 | 33.827795,28.065129,30.315809 |
| 36 | 28.789703,32.245184,41.21117 |

Each row contains the execution time result, in seconds, from a SQL block, and the 3 comma separated numbers represent the execution time of3 queries in that block.

Row 1 is the execution result of block 1, which contains query 1, 2, 3, in that order

Row 2 is the execution result of block 2, which contains query 2, 3, 1, in that order

Row 3 is the execution result of block 3, which contains query 3, 1, 2, in that order

Then, row 4, 5, 6 repeat that pattern: execution of block1, block 2, block 3, in that order.

Then, row 7, 8, 9, repeat that pattern again, etc.

For results from block 1, 3 query time from query 1, query 2, query 3 are in order.

For results from block 2 and block 3, I need to switch the order in a row of results to keep the order of results from query 1, query 2, query 3, in that order.

So, for each row, the first number is always the time to execute query 1, and the second number is always the time to execute query 2, and the third number is always the time to execute query 3.

# ANOVA Analysis

A screenshot of a computer

Description automatically generated

First, I did anova analysis to compare the mean of query 1 to that of query 2 and 3.

Then, I did anova analysis to compare the mean of query 2 to that of query 1, and 3.

Then, I did anova analysis to compare the mean of query 3 to that of query 1 and 2.

It appears that the

1. (query 1, query 2), (query 2, query 3), (query 3, query 1) all may have significant differences.
2. But differences between (query 1, query 2) may be more significant

Then I did another analysis to verify the above comparisons. This time, I used t test

To check the means of individual query to the average of means of all 3 queries:

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

From the analysis, it appears that none of the means of individual queries

are significantly differ from the average of all means of all 3 queries.

To be sure, I did the third batch of analysis. This time, I used t test to do pairwise analysis

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

From these result, it verified that there is no any significant difference in means of execution time between any two of these 3 queries

# 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 $$;