Experiment No.1

Title: Execution of Parallel Database queries.

Batch: A2 **Roll No.: 16010421059** **Experiment No.: 1**

Aim: To execute Parallel Database queries.

Resources needed: PostgreSQL 9.3

Theory

A parallel database system seeks to improve performance through parallelization of various

operations, such as loading data, building indexes and evaluating queries. Although data may be stored in a distributed fashion, the distribution is governed solely by performance considerations. Parallel databases improve processing and input/output speeds by using

multiple CPUs and disks in parallel. Centralized and client–server database systems are not powerful enough to handle such applications. In parallel processing, many operations are

performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.
Types of parallelism :

• Interquery parallelism: Execution of multiple queries in parallel

• Interoperation parallelism: Execution of single queries that may consist of more than one operations to be performed.

- Independent Parallelism - Execution of each operation individually in different

  processors only if they can be executed independent of each other. For example, if we need to join four tables, then two can be joined at one

  processor and the other two can be joined at another processor. Final join can be done later.

- Pipe-lined parallelism - Execution of different operations in pipe-lined

  fashion. For example, if we need to join three tables, one processor may join two tables and send the result set records as and when they are produced to the

  other processor. In the other processor the third table can be joined with the incoming records and the final result can be produced.

• Intraoperation parallelism Execution of single complex or large operations in parallel in

multiple processors. For example, ORDER BY clause of a query that tries to execute on millions of records can be parallelized on multiple processors.

Procedure:

Parallel queries provide parallel execution of sequential scans, joins, and aggregates etc.

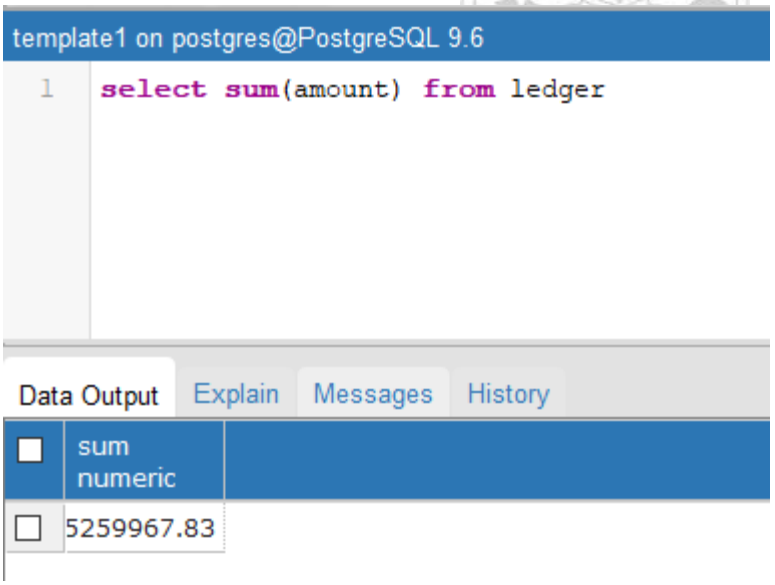
Parallel queries provide parallel execution of sequential scans, joins, and aggregates. To make the performance gains need a lot of data.

```
create table ledger (
  id serial primary key,
  date date not null,
  amount decimal(12,2)  not null
);
insert into ledger (date, amount)
  select current_date - (random() * 3650)::integer,
  (random() * 1000000)::decimal(12,2) - 50000
  from generate_series(1,50000000);
```

Explain analyse select sum(amount) from ledger;

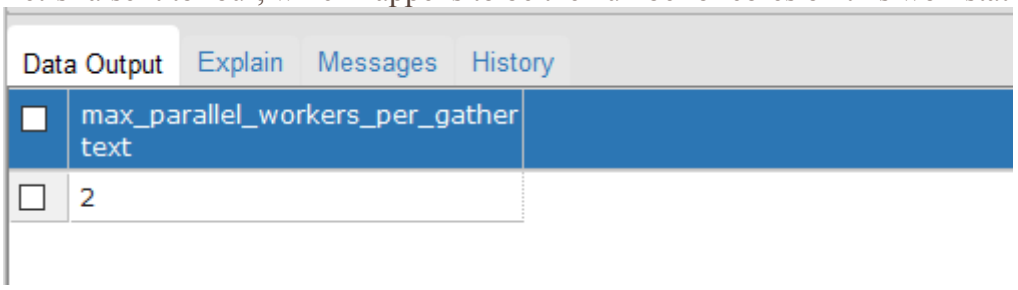
Reading the output, we can see that Postgres has chosen to run this query sequentially.

Parallel queries are not enabled by default. To turn them on, we need to increase a config param called max_parallel_workers_per_gather.



- **Show max_parallel_workers_per_gather;**

Let's raise it to four, which happens to be the number of cores on this workstation.



- **Set max_parallel_workers_per_gather to 4;**

Explaining the query again, we can see that Postgres is now choosing a parallel query.
And it's about four times faster.

template1 on postgres@PostgreSQL 9.6

```
1  Set max_parallel_workers_per_gather to 4;
2
3  Show max_parallel_workers_per_gather;
```

Data Output | Explain | Messages | History

max_parallel_workers_per_gather text
4

Explain analyse select sum(amount) from ledger;

```
6  Explain analyse select sum(amount) from ledger;
```

Data Output | Explain | Messages | History

QUERY PLAN text
Finalize Aggregate (cost=47569...
-> Gather (cost=475697.27..475...
Workers Planned: 4
Workers Launched: 4
-> Partial Aggregate (cost=4746...
-> Parallel Seq Scan on ledger (c...
Planning time: 0.064 ms
Execution time: 8281.489 ms

Tue Jan 17 2023 14:38:35 GMT+0530 (India Standa... | select sum(amount) from ledger; | 1 | 8 secs

The planner does not always consider a parallel sequential scan to be the best option. If a query is not selective enough and there are many tuples to transfer from worker to worker, it may prefer a "classic" sequential scan. PostgreSQL optimises the number of workers according to size of the table and the min_parallel_relation_size.

Similar ways we can execute join operation and check parallel execution of sequential join.

template1 on postgres@PostgreSQL 9.6

```
1  Set max_parallel_workers_per_gather to 3;
2
3  Explain analyse select sum(amount) from ledger;
4
5
6
7
```

Data Output | Explain | Messages | History

QUERY PLAN text
Finalize Aggregate (cost=52106...
-> Gather (cost=521060.24..521...
Workers Planned: 3
Workers Launched: 3
-> Partial Aggregate (cost=5200...
-> Parallel Seq Scan on ledger (c...
Planning time: 0.082 ms
Execution time: 8570.690 ms

Tue Jan 17 2023 14:52:30 GMT+0530 (India Standa... | Explain analyse select sum(amount) from ledger; | 8 | 9 secs

CREATED TABLE:

template1 on postgres@PostgreSQL 9.6

```
1  CREATE TABLE head(
2      id serial,
3      dept_id int,
4  );
6  CREATE TABLE dept(
7      dept_id serial,
8      profit int,
9      );
```

INSERTED VALUES:

```
insert into dept(dept_id,profit) select floor(random()*123+1::numeric),floor(random()*(1-100)+100::numeric)
from generate_series(1,10000);
```

DISPLAY:

-

```
31  select * from dept
```

Data Output | Explain | Messages

profit integer	dept_id integer
80	12
50	31
78	97
2	103
8	106
94	18
48	61
74	70
43	34
90	33
71	110

id integer	dept_id integer
50002	50
50003	113
50004	75
50005	65
50006	88
50007	122
50008	66
50009	74
50010	6
50011	1
50012	106

explain analyse select library1.id,library1.quantity,library2.location from library2, library1 where library1.id=library2.id;

```
33  explain analyse select head.id,dept.profit from dept,
34  head where head.dept_id=dept.dept_id;
```

Data Output | Explain | Messages | History

QUERY PLAN text
Hash Join (cost=270.00..9927.1...
Hash Cond: (dept.dept_id = hea...
-> Seq Scan on dept (cost=0.00....
-> Hash (cost=145.00..145.00 r...
Buckets: 16384 Batches: 1 Mem...
-> Seq Scan on head (cost=0.00...
Planning time: 0.225 ms
Execution time: 110.744 ms

SET max_parallel_workers_per_gather TO 3;

```
35  SET max_parallel_workers_per_gather TO 3;
```

Data Output | Explain | Messages | History

```
SET

Query returned successfully in 328 msec.
```

```
35  SET max_parallel_workers_per_gather TO 1024;
```

Data Output | Explain | Messages | History

```
SET

Query returned successfully in 672 msec.
```

Questions:

1. Explain the parallelism achieved in the experiment you performed.
   Ans. parallelism is a different kind of parallelism that, instead of relying on process or task concurrency, is related to both the flow and the structure of the information. An analogy might revisit the automobile factory from our example in the previous section.

2. With comparison of the results explain how degree of parallelism (no of parallel processors) affect the operation conducted.

Ans. .in order to the degree of parallelism affects the output by reducing its time to execute and this helping to
run by providing extra processor which help to execute the process more smoothly and efficiently. Typically
a computer scientist will divide a complex task into multiple parts with a software tool and assign each part
to a processor, then each processor will solve its part, and the data is reassembled by a software tool to read
the solution or execute the task.
Typically each processor will operate normally and will perform operations in parallel as instructed, pulling
data from the computer's memory. Processors will also rely on software to communicate with each other so
they can stay in sync concerning changes in data values. Assuming all the processors remain in sync with one
another, at the end of a task, software will fit all the data pieces together

Results: (Program printout with output)

Outcomes:

CO1: Design advanced database systems using Parallel, Distributed and In-memory databases and its implementation.

References:

Conclusion: (Conclusion to be based on the outcomes achieved)

Books/ Journals/ Websites:

Executed Parallel Database queries Successfully.

1. Elmasri and Navathe, "Fundamentals of Database Systems", Pearson Education
2. https://www.postgresql.org/docs/

Grade: AA / AB / BB / BC / CC / CD /DD