Journal of Sisfotek Global

ISSN (Online): 2721 - 3161, ISSN (Print): 2088 – 1762
DOI: http://dx.doi.org/10.38101/sisfotek.v12i1.477
Vol. 12, No. 1, March 2022, pp. 50-57

Implementation of Database Distributed Sharding Horizontal Partition in MySQL. Case Study of Application of Food Serving On Kemkes

Samidi[1], Ronal Yulyanto Suladi[2] & Ario Bambang Lesmana[3]

[1,2,3]Universitas Budi Luhur, Jakarta, Indonesia, 12270
[1]samidi@budiluhur.ac.id, [2]surel.ronal@gmail.com, [3]ariobambanglesmana@gmail.com

ARTICLE HISTORY

Received : 8 March, 2022
Revised : 21 March, 2022
Accepted : 29 March, 2022

KEYWORDS

Single database
Database distribution
Mariadb
Spider Storage Engine
Sharding

ABSTRACT

In today's digital era, database systems are becoming a mandatory thing that is used in every industry field to improve the performance of the industry, over time with the increasing need for databases, single databases and single servers are considered less capable in handling data that continues to increase in number, so it often leads to a decline in performance. Therefore came the idea of centralized data storage that can be modified to accommodate the availability, scalability, reliability, and management of data. One way that can be done is to apply Distribute Database using Sharding Horizontal Partition technique and experimental methods using mariadb and spider engine. The test was carried out on the database of food processing sites (TPP) in the TPP application at the ministry of health. The test results prove that although in some journals about Sharding, more sharding is implemented on NoSQL DBMS such as MongoDB, influx dB. In this study it can be proven that sharding can also be done in relational DBMS such as MariaDB and MySQL, data can be distribute and load process too, test result successful although the performance of time query response is still better single database than distribute database.

1. Introduction

In today's digital age, database systems are used in every need for data. The idea of centralized data storage can be modified to accommodate data availability, scalability, reliability, and management; of course, higher performance and lower costs become a challenge. It can be adjusted by distributing data using sharding against the database. The sharded database architecture consists of multiple server nodes deployed across the server using network infrastructure and distribution of data allocation to various server nodes [1]. By utilizing sharding databases or horizontal partitions, databases are divided into smaller pieces or fractions across multiple data nodes in a cluster. Each node with data is distributed into a subset and responsible to itself.

The Food Processing Site Database (TPP) is a data set used by TPP Information system which is a normal procedure where data is collected and then processed into a information that can be distributed to users, can obtain transaction management needs, support the organization's operations, are managerial in a strategic activity of the organization and provide certain parties with various kinds of reports that needed [2]. The TPP data includes restaurants, restaurants catering, Boga services, drinking water depots, canteens, and other food supply and processing fields. Outcome TPP application is of them sertifikat laik higienes for provider of catering services, food, sanitation [3]. The growing need for supervision and coaching of TPP in each province often causes problems in the form of decreased system performance and more excellent database storage with a single server and single database.

A distributed computing system consists of several processing nodes interconnected in a computer network and cooperating in performing certain given tasks. In general, the distribution of compounds partitions large parts into smaller pieces and resolves them efficiently in a coordinated manner. In theory, database distribution using sharding methods can be done in relational and nonrelational DBMS/NoSQL [4].

NoSQL which is an unstructured database is widely used as a database storage time series, for example InfluxDB [5]

In this research, we use MariaDB relational Database Management System (RDBMS) [6] based on open-source data placement will be distributed to 3 (three) servers run using containers. In Container using linux we can running nodes server with light resource and increase perform [7].

There are several previous studies such as Afri Darwis [8] in his research title Implementation of Distributed Database Using Mysql At PT Thamrin Brothers Palembang, the issue discussed is How to Design and Implement Distributed Databases on PT Thamrin Brothers Palembang, using action research methods and decision-makers, as a result of distributed databases used in PT Thamrin Brothers Palembang where the database is integrated. Distributed between branches throughout South Sumatra, and with the design of this database, PT Thamrin Brothers can develop it for the operational benefit of the company.

In Previous research distributed database implemented using heterogenous method using MySQL and Oracle XE, with database link and ODBC Driver as communication between databases [9].

While Bryant Plaudo Santoso, Agustinus Noertjahyana, and Justinus Andjarwirawan [10] in his journal entitled Implementation of Distributed Database On Learning Management System Using Redhat Openshift Platform, the problem discussed is scalable on the web server, If only the webserver is scalable and only connected to a database server, then it is feared that there will be bottlenecks, with the *Sharding* method, the result of transaction per second or success rate becomes better.

In Grech research Designing and Implementing a Distributed Database for a Small Multi-Outlet Business, results stated that the distribution database is a database development that has advantages in sharing and replicating data between databases [11].

Next, Naoyuki Miyamoto, Ken Higuchi, and Tatsuo Tsuji [12], with their journal Incremental Data Migration for Multi-Database Systems Based on MySQL with SPIDER Storage Engines, tried to solve the problem of performance decrement from load imbalance among individual databases using incremental data migration methods, the result is a total data migration technique effective for high execution times (except for some situations).).

The difference between this research and previous is that this study uses MariaDB Relational Database Management System (RDBMS) with an engine that is a plugin of the RDBMS, namely, spider storage engine, or in another sense, it can be said that the database distribution process occurs at the database level at the coding level in the application. In previous research, Santoso ran the MySQL cluster as an engine of the DBMS on the OpenShift platform, which is used as a database of learning management systems with the results of research that leads to the MySQL cluster's success rate of the database is getting better. Darwis, in previous research, discussed the design of distributed databases by using MySQL as a DBMS and conducting replication techniques for distribution processes to several branches, as well as primary visual applications as their frontend with the results of the creation of a DFD database distribution design where each transaction data process will be marked with branch ID as the purpose of storage of the database. Miyamoto, in previous research, used MySQL and spider storage engines as a solution to carry incremental migration data into multi databases so that as the database grows more extensive and can operate without downtime by dividing large amounts of migration data into smaller pieces of data, the results of the study that with a small number of sub-data inserted into multi-database will be more accessible and have a good execution time value. Better, and that's without interfering with the existing operations of the running database.

Based on the background above, the problems that can be formulated are:

1. How the process of migrating the database from a single server to 3 server regions, How to picture networking and addressing each server, How the process of configuration of container servers includes operating systems, DBMS, computer networks according to network topology
2. How to configure sharding with the horizontal partition database server using Mariadb spider engine storage
3. How to configure connections, user databases, and privileges in each server node
4. How does the comparison between a single database server compare to after the database server is distributed to three servers, as well as how the test results exist.

This research was conducted by taking data on the transaction table of tt_tpm that have a large number of rows and distributed to several RDBMS by sharding / horizontal partition method according to the parameters of the attributes kd_prov done manually; in this research, server infrastructure uses docker containers that run RDBMS MariaDB image in each container. The data used is a table of tt_tpm transactions taken from a single database server with the tpm_db.

Related to this, the purpose of this study is to plan testing to improve server performance by adding server infrastructure in several regions virtually. Database server workload sharing is divided into areas covering Western Indonesia, Central Indonesia, and Eastern Indonesia. For the abo planning, we tried to implement the application of distributed databases with the sharding horizontal partition method. The table in the database has many rows in the party into sections based on the criteria of provincial code (zone region).

Implementing distributed databases and sharing data placement and storage also allows users with certain provinces to directly access and make transactions on servers within the region to make the

TPP application a scalable application in overcoming the amount of data that develops with a distributed database.

The purpose of this research is:

1. Proving database distribution theory using the sharding horizontal partition method in the RDBMS.
2. Implementation of data storage set (table) centralized database (single server) to distributed (3 servers)
3. Prove query execution performance using a single database server by distributing the database server

Distributed databases self-transmitted database is a collection of several logically interrelated databases distributed over a computer network. A database management system (DDBMS) is a software system that manages distributed databases while making distribution transparent to users.

To meet the conditions of a distributed database, at least meet the following requirements:

- Connection of database nodes over a computer network. There are several computers, called nodes. These nodes must be connected to the underlying network to transmit data and commands between nodes.
- The logical interrelationship of the connected essential The information in the various database nodes must be logically related.
- There is a possibility of homogeneity among connected nodes. Not all nodes need to be identical in data, hardware, and software [13].

In a distributed database, database information is allocated and stored in corresponding nodes, according to partition or fragment or called horizontal fragmentation or sharding. A so-called technique can be used to partition each relation based on a particular attribute (e.g., department and others) [14]

A horizontal fragment or fraction of a relation is a condition that can determine a subset of the tuple in that r is included in flat fragments on one or more relationship attributes or by some other mechanism. Often, only one point is involved in the state [15].

In another sense, sharding is a database architecture pattern that separates one table into several different tables, commonly known as partitions. Each partition has the same schema and columns but also different rows. The data stored in each partition is also unique and independent of the data stored on other partitions Sharding consists of 2 horizontal partitioning and vertical partitioning. Examples of data sharing using sharding can be seen in the image below. Horizontal partitioning divides a table into multiple tables; each table has the same number of columns as the original table but has smaller rows. Vertical partitioning is a partition method that divides tables into tables with the same number of rows as the original table but fewer columns [16].

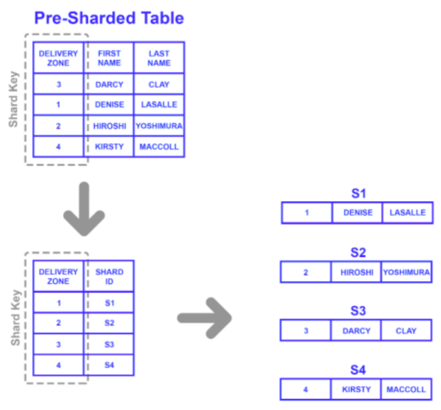


Figure 1: Sharding Horizontal Partition by list

Database sharding is one way to scaling-out (horizontally) in spreading and placing the burden of data processes to several database server nodes and dividing the data traffic process to be more optimal. Scaling-out in practice is adding the number of server nodes. By scaling out, data will be distributed. Storage media and computing resources will be more flexible in scaling (sizing) against database needs, number of server nodes, number of storage media, and computing sources in the future.

MySQL and MariaDB are database management systems developed by Oracle. One of the essential features is that users can choose a storage engine according to their needs. For example, users can use a memory storage engine to access the database quickly.

A spider storage engine is one of the MySQL storage engines developed by Kentoku Shiba. It does not have record data and only references other databases using table links. In other words, a spider storage engine is a collection of connections such as symbolic links in the UNIX operating system. The records are fermented not only table links in the same computer but also other computer table links. Users can access tables on multiple computers as a single table by using SQL statements without specific parameters. Spider Storage Engine is not the default engine of MySQL or MariaDB; it is necessary to add module packages to the DBMS.

Spider storage engines define tables while retaining the original engine from the table. For example, link tables defined using the InnoDB storage engine have the same capabilities as InnoDB, namely record-level and transactional lock capabilities.

Spider storage [17] is one of the sharding and proxy solutions that can be used to federate tables from MariaDB MySQL/Oracle server nodes as in local servers, and spiders can create sharding databases by using the partition table feature.

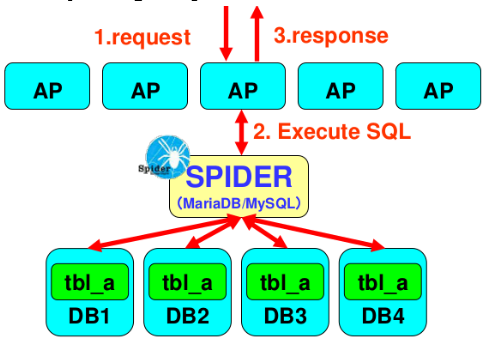


Figure 2: Spider sharding architecture uses partition tables based on rules

2. Methods

This study used experimental methods to implement distributed databases, using sharding with spider engine storage than will be tested several SQL statements into the database.

The dataset used is the EMONEV Kemkes application table database in 2021; the table used is a table of tt_tpm numbering 250,000; there will be two databases in a single server and a database in 4 containers representing four different servers. The data used using the data retrieval method is a sampling from the TPP application database.

The research stage is :

1. Taking the data set of tables tt_tpm 2021
2. Then installing and configuring the Linux operating system,
3. Create docker container
4. Installing and configuring the MariaDB RDBMS,
5. Installing additional spider storage engine packages,
6. Creating new databases by sharding,
7. Creating table links, creating users in each node to allow access to server spiders,
8. Inserting datasets into spider servers,
9. Perform a select query on a server spider.

3. Result and Discussions

TPP application is an information system used for monitoring and evaluation (MONEV) of food processing places; this application is used by puskemas, municipal district health office, local health service, and central health service. The operator of this application is a health center that numbers 10,230 puskesmas throughout Indonesia. Data is input to the application after the IKL form (Extension) is filled through visiting the TPP location. TPP applications with a national scale of use infrastructure are on 1 (one) server for Webserver and Database services.

Migrating Database and topologies desgin

The TPP application database has a tt_tpm table containing TPP information data registered in the ministry for coaching and hygienic certification; in this study, we shard the tt_tpm table written in this study distributed into three containers servers.
Here is the topology of the existing network with one node database server :

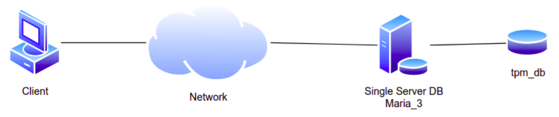


Figure 3: Single Database Server Topology (Existing)

There is a MariaDB DBMS service in a single server database, with 1 (one) database and supporting tables of TPP applications; this single database does not use partition tables; it only uses indexes on tables.
Here is the topology of the network with one spider server and three nodes with three shardings :

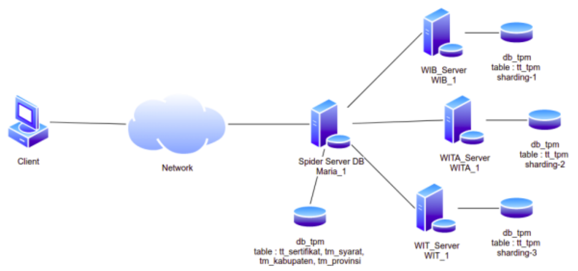


Figure 4: Prototype-1 Distribute Database Implementation Topology

There is a MariaDB DBMS service in the distribution database server, with 1 (one) spider server database and three server nodes that will be the place of database allocation according to the partition (3 sharding) specified in the server spider. The server spider runs the MARIADB DBMS and a collection of link tables to access the server nodes. In other words, each node is allocated to 1 partition according to that node zone.

Here is the topology of the network with one spider server and three nodes with six shardings :

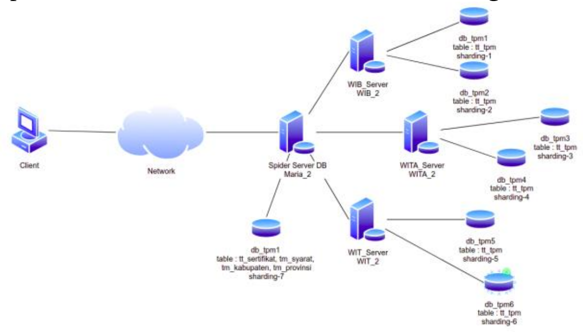


Figure 5: Prototype-2 Distribute Database Implementation Topology

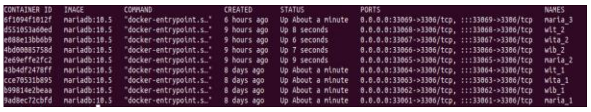
In this research, the implementation of the distributed database server, there is a MariaDB DBMS service, with 1 (one) spider server database and three server nodes that will be the place of database allocation according to the partition (6 sharding) specified in the server spider. In addition, the server spider is also sharding to the local DBMS for supporting tables in partitions, namely: tables tt_sertifikat, tm_syarat, tm_kabupaten, tm_provinsi. The server spider runs the MARIADB DBMS and a collection of link tables to access the server nodes. In other words, each node is allocated to 2 partitions according to that node zone.

Here is a list of IP addresses for simulations in this study, are as follows:

No.	Host	IP Address	Keterangan	Group
1	Maria_1 - Server Head Office	172.18.0.11	Spider Server-Test 1	Prototype-1
2	Wib_1 - Server Zona Barat	172.18.0.12	Node WIB-Test 1	
3	Wita_1 - Server Zona Tengah	172.18.0.13	Node WITA-Test 1	
4	Wit_1 - Server Zona Timur	172.18.0.14	Node WIT-Test 1	
5	Maria_2 - Server Head Office	172.18.0.21	Spider Server-Test 2	Prototype-2
6	Wib_2 - Server Zona Barat	172.18.0.22	Node WIB-Test 2	
7	Wita_2 - Server Zona Tengah	172.18.0.23	Node WITA-Test 2	
8	Wit_2 - Server Zona Timur	172.18.0.24	Node WIT-Test 2	
9	Maria_3 - Server Single DB	172.18.0.31	Node SingleDB	Single Server DB

Figure 6: Tabel of IP Address Server

List of containers running in research :


```
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS                 PORTS                                           NAMES
6f1094f1012f   mariadb:10.5   "docker-entrypoint.s…"   6 hours ago    Up About a minute      0.0.0.0:33069->3306/tcp, :::33069->3306/tcp     maria_3
d551053a60ed   mariadb:10.5   "docker-entrypoint.s…"   9 hours ago    Up 8 seconds           0.0.0.0:33068->3306/tcp, :::33068->3306/tcp     wit_2
e088e13bb6b9   mariadb:10.5   "docker-entrypoint.s…"   9 hours ago    Up 6 seconds           0.0.0.0:33067->3306/tcp, :::33067->3306/tcp     wita_2
4bd00085758d   mariadb:10.5   "docker-entrypoint.s…"   9 hours ago    Up 7 seconds           0.0.0.0:33066->3306/tcp, :::33066->3306/tcp     wib_2
2e69effe2fc2   mariadb:10.5   "docker-entrypoint.s…"   9 hours ago    Up 9 seconds           0.0.0.0:33065->3306/tcp, :::33065->3306/tcp     maria_2
43b4df2478ff   mariadb:10.5   "docker-entrypoint.s…"   8 days ago     Up About a minute      0.0.0.0:33064->3306/tcp, :::33064->3306/tcp     wit_1
cce70531b895   mariadb:10.5   "docker-entrypoint.s…"   8 days ago     Up About a minute      0.0.0.0:33063->3306/tcp, :::33063->3306/tcp     wita_1
b99814e2beaa   mariadb:10.5   "docker-entrypoint.s…"   8 days ago     Up About a minute      0.0.0.0:33062->3306/tcp, :::33062->3306/tcp     wib_1
9ad8ec72cbfd   mariadb:10.5   "docker-entrypoint.s…"   8 days ago     Up About a minute      0.0.0.0:33061->3306/tcp, :::33061->3306/tcp     maria_1
```
Figure 7: Container List

Sharding uses a horizontal partition based on the provincial code attribute (KD_PROV), following the contents of the local code field in the tt_tpm table by region zone. Data with a specific provincial codespecificelocal the specified server node based on this sharding.,

We made a prototype-1 consisting of 1 spider server three server nodes consisting of 3 sharding (each node one sharding), here is the allocation of sharding:

11 - Aceh	12 - Sumatera Utara	13 - Sumatera Barat	14 - Riau
15 - Jambi	16 - Sumater Selatan	17 - Bengkulu	18 - Lampung
19 - Bangka Belitung	21 - Kep Riau	31 - DKI Jakarta	32 - Jawa Barat
33 - Jawa Tengah	34 - DI Yogyakarta	35 - Jawa Timur	36 - Banten

0 - SUBMIT Karkes	51 – Bali	52 - NTB	53 - NTT
61 - Kalimantan Barat	62 - Kalimantan Tengah	63 - Kalimantan Selatan	64 - Kalimantan Timur
65 - Kalimantan Utara			

71 - Sulawesi Utara	72 - Sulawesi Tengah	73 - Sulawesi Selatan	74 - Sulawesi Tenggara
75 - Gorontalo	76 - Sulawesi Barat	81 - Maluku	82 - Maluku Utara
91 - Papua Barat	94 – Papua		

Figure 8: Prototype-1

Then we also made a prototype-2 consisting of 1 spider server three server nodes consisting of 6 shardings (each node two sharding) the goal is to minimize sharding and the number of row data on the table tt_tpm per sharding, here is the allocation of sharding :

11 – Aceh	12 - Sumatera Utara	13 - Sumatera Barat	14 - Riau
15 - Jambi	16 - Sumater Selatan	17 - Bengkulu	18 - Lampung

19 - Bangka Belitung	21 - Kep Riau	31 - DKI Jakarta	32 - Jawa Barat
33 - Jawa Tengah	34 - DI Yogyakarta	35 - Jawa Timur	36 - Banten

0 - SUBMIT Karkes	51 - Bali	52 - NTB	53 - NTT

61 - Kalimantan Barat	62 - Kalimantan Tengah	63 - Kalimantan Selatan	64 - Kalimantan Timur
65 - Kalimantan Utara			

71 - Sulawesi Utara	72 - Sulawesi Tengah	73 - Sulawesi Selatan	74 - Sulawesi Tenggara
75 - Gorontalo			

6 - Sulawesi Barat	81 - Maluku	82 - Maluku Utara	91 - Papua Barat
94 - Papua			

Figure 9: Prototype-2

Configure Sharding

This step explain configure sharding :

1. Installation Database engine
   In linux operating system we can use this command for install mariadb database :
   apt install mariadb-server mariadb-client
2. Create scheme sharding database db_tpm and table tt_tpm [18]

```
CREATE SCHEMA db_tpm DEFAULT CHARACTER SET utf8;
CREATE TABLE db_tpm.tt_tpm (
  id int(11) NOT NULL AUTO_INCREMENT,
  nama_tpm varchar(255) NOT NULL,
  kd_prov varchar(2),
  kd_kab varchar(4),
  kd_kec varchar(7),
  kd_kel varchar(10) NOT NULL DEFAULT '0',
  id_puskesmas varchar(11),
  id_area int(11) NOT NULL DEFAULT '0',
  id_wilker int(11) NOT NULL DEFAULT '0',
  alamat varchar(255),
  nama_pengusaha varchar(255),
  jumlah_karyawan int(11),
  id_jns_tpm int(11),
  total_point int(11),
  status_tpm varchar(2),
  last_date_scoring date,
  init_status_layak int(11),
  lat varchar(20) NOT NULL,
  lng varchar(20) NOT NULL,
  `desc` varchar(255) NOT NULL,
  status_sertifikat int(11),
    create_by varchar(225),
    create_date datetime,
    update_by varchar(225),
    update_date datetime,
    delete_status enum('0', '1') DEFAULT '0',
    delete_by varchar(225),
    delete_date datetime,
    PRIMARY KEY (id,kd_prov),
    KEY id (id)
) ENGINE=spider comment='database "db_tpm",table "tt_tpm"'
PARTITION BY LIST columns (kd_prov) (
  PARTITION wib_tpm_partition VALUES IN ('11','12','13','14','15','16','17','18','19','21','31','32','33','34','35','36')
COMMENT = 'srv "wib_db_tpm"' ENGINE = SPIDER,
  PARTITION wit_tpm_partition VALUES IN ('71','72','73','74','75','76','81','82','91','94')
COMMENT = 'srv "wit_db_tpm"'ENGINE = SPIDER,
  PARTITION wita_tpm_partition VALUES IN ('0','51','52','53','61','62','63','64','65')
COMMENT = 'srv "wita_db_tpm"' ENGINE = SPIDER
      );
```

Configure connection and spider user privileges

1. Create Server database parameter
   For Instance in wib server, configure parameter server:

   ```
   CREATE SERVER wib_db_tpm
   FOREIGN DATA WRAPPER mysql
   OPTIONS (
     HOST '172.18.0.12',
     PORT 3306,
     USER 'spider_user',
     PASSWORD 'password',
     DATABASE 'db_tpm'
           );
   ```
2. Create user for spider user to access all nodes

   ```
   create user 'spider_user'@'%' identified by 'password';
   grant all privileges on *.* to 'spider_user'@'%' with grant option;
   flush privileges;
   ```

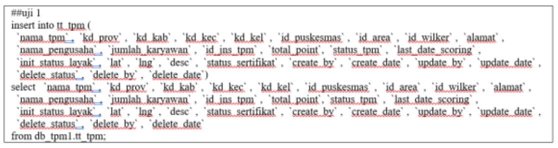
Comparison between single database and three database server

In this study, there are several queries to prove that data can be stored in allocations on different server nodes based on partition attributes specified according to the table above.

The insert query inserts from select to the table tt_tpm, then inserted into the table tt_tpm in the database in the server spider. The amount of data inserted is 250,000 rows.

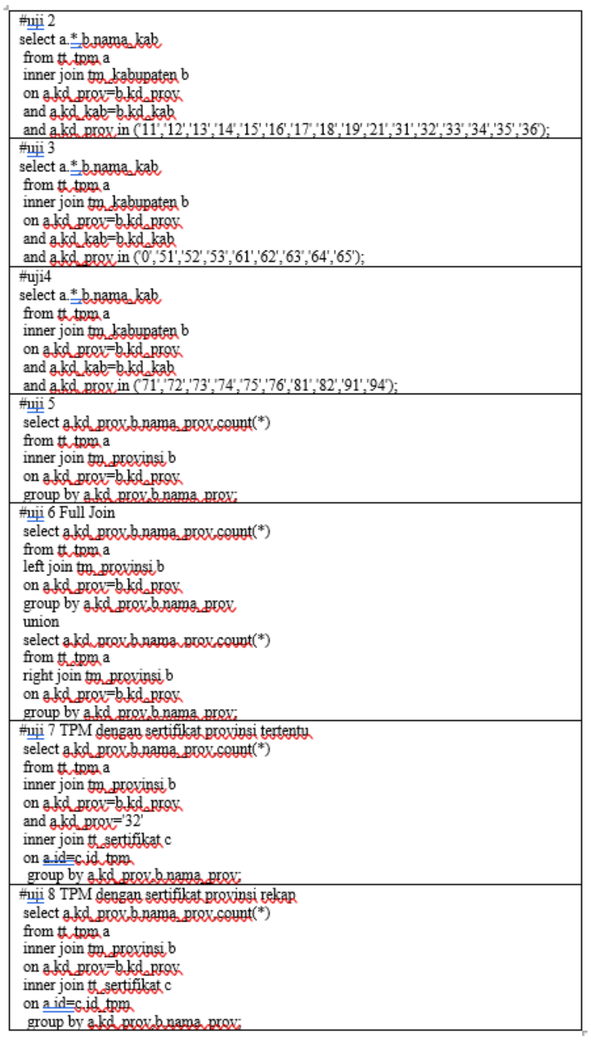
The next test is a select query to see the response time of query execution, which is carried out on a single server, prototype-1, and prototype-2 according to the table above. The queries used for testing are as follows :

The insert query inserts from select to the table tt_tpm, then inserted into the table tt_tpm in the database in the server spider. The amount of data inserted is 250,000 rows.



```
##uji 1
insert into tt_tpm (
 `nama_tpm`, `kd_prov` , `kd_kab` , `kd_kec` , `kd_kel` , `id_puskesmas` , `id_area` , `id_wilker` , `alamat` ,
 `nama_pengusaha`, `jumlah_karyawan` , `id_jns_tpm` , `total_point`, `status_tpm` , `last_date_scoring` ,
 `init_status_layak`, `lat` , `lng` , `desc` , `status_sertifikat` , `create_by` , `create_date` , `update_by` , `update_date` ,
 `delete_status`, `delete_by` , `delete_date`)
select `nama_tpm`, `kd_prov` , `kd_kab` , `kd_kec` , `kd_kel` , `id_puskesmas` , `id_area` , `id_wilker` , `alamat` ,
 `nama_pengusaha`, `jumlah_karyawan` , `id_jns_tpm` , `total_point`, `status_tpm` , `last_date_scoring` ,
 `init_status_layak`, `lat` , `lng` , `desc` , `status_sertifikat` , `create_by` , `create_date` , `update_by` , `update_date` ,
 `delete_status`, `delete_by` , `delete_date`
from db_tpm1.tt_tpm;
```

The next test is a select query to see the response time of query execution, which is carried out on a single server, prototype-1, and prototype-2 according to the table above. The queries used for testing are as follows:



```
#uji 2
select a.*,b.nama_kab
 from tt_tpm a
 inner join tm_kabupaten b
 on a.kd_prov=b.kd_prov
 and a.kd_kab=b.kd_kab
 and a.kd_prov in ('11','12','13','14','15','16','17','18','19','21','31','32','33','34','35','36');

#uji 3
select a.*,b.nama_kab
 from tt_tpm a
 inner join tm_kabupaten b
 on a.kd_prov=b.kd_prov
 and a.kd_kab=b.kd_kab
 and a.kd_prov in ('0','51','52','53','61','62','63','64','65');

#uji4
select a.*,b.nama_kab
 from tt_tpm a
 inner join tm_kabupaten b
 on a.kd_prov=b.kd_prov
 and a.kd_kab=b.kd_kab
 and a.kd_prov in ('71','72','73','74','75','76','81','82','91','94');

#uji 5
 select a.kd_prov,b.nama_prov,count(*)
 from tt_tpm a
 inner join tm_provinsi b
 on a.kd_prov=b.kd_prov
 group by a.kd_prov,b.nama_prov;

#uji 6 Full Join
 select a.kd_prov,b.nama_prov,count(*)
 from tt_tpm a
 left join tm_provinsi b
 on a.kd_prov=b.kd_prov
 group by a.kd_prov,b.nama_prov
 union
 select a.kd_prov,b.nama_prov,count(*)
 from tt_tpm a
 right join tm_provinsi b
 on a.kd_prov=b.kd_prov
 group by a.kd_prov,b.nama_prov;

#uji 7 TPM dengan sertifikat provinsi tertentu
 select a.kd_prov,b.nama_prov,count(*)
 from tt_tpm a
 inner join tm_provinsi b
 on a.kd_prov=b.kd_prov
 and a.kd_prov='32'
 inner join tt_sertifikat c
 on a.id=c.id_tpm
  group by a.kd_prov,b.nama_prov;

#uji 8 TPM dengan sertifikat provinsi rekap
 select a.kd_prov,b.nama_prov,count(*)
 from tt_tpm a
 inner join tm_provinsi b
 on a.kd_prov=b.kd_prov
 inner join tt_sertifikat c
 on a.id=c.id_tpm
  group by a.kd_prov,b.nama_prov;
```

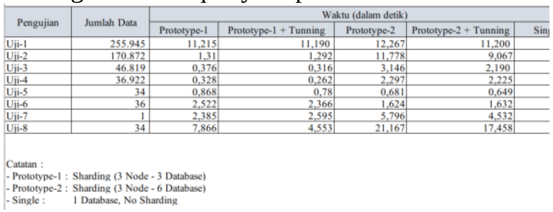
Figure 10: Testing script SQL

After testing, here are some results from the test:

1. By using a distributed spider storage engine database can be done, either by federate and or by sharding, in this research, we use sharding / horizontal partition against a table with a certain

number of rows, which is partitioned with specific conditions/rules, in this case, using provincial code (can be viewed above). As a result, each partition table can occupy storage on each server node according to local code rules.

2. Spider engine storage, the partitioned table remains with the original engine. We can perform the Data Manipulation Language (DML) function through the spider server node and the database server node.
3. CPU load usage and memory can be distributed properly
4. Testing for time query responses is as follows:

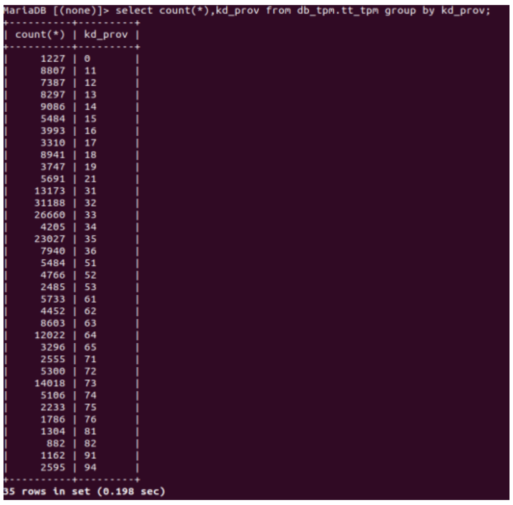


Pengujian	Jumlah Data	Waktu (dalam detik)				
		Prototype-1	Prototype-1 + Tunning	Prototype-2	Prototype-2 + Tunning	Sin
Uji-1	255.945	11,215	11,190	12,267	11,200	
Uji-2	170.872	1,31	1,292	11,778	9,067	
Uji-3	46.819	0,376	0,316	3,146	2,190	
Uji-4	36.922	0,328	0,262	2,297	2,225	
Uji-5	34	0,868	0,78	0,681	0,649	
Uji-6	36	2,522	2,366	1,624	1,632	
Uji-7	1	2,385	2,595	5,796	4,532	
Uji-8	34	7,866	4,553	21,167	17,458	

Catatan :
- Prototype-1 : Sharding (3 Node - 3 Database)
- Prototype-2 : Sharding (3 Node - 6 Database)
- Single : 1 Database, No Sharding

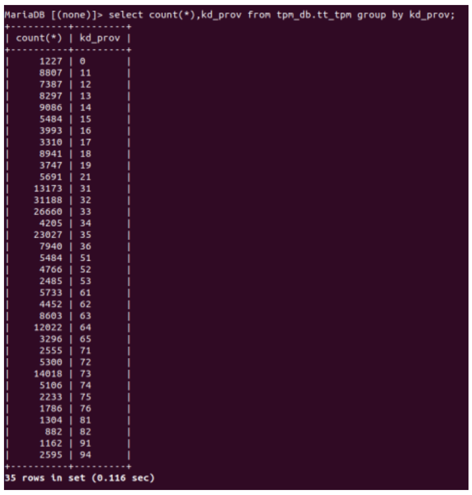
Figure 11: Result testing

5. The following screenshot of the database distribution based on the provincial code partition can be seen from the results of the following query :



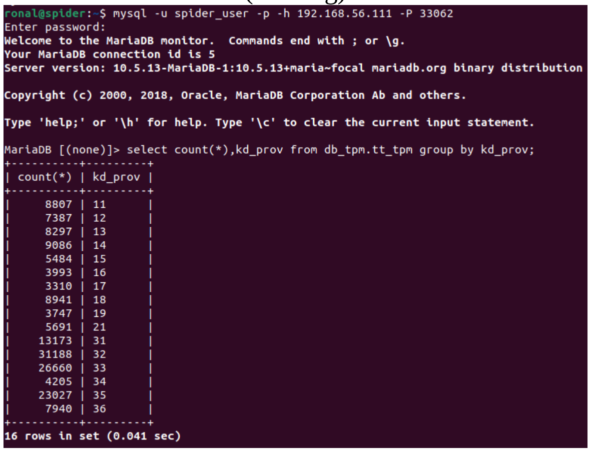
```
MariaDB [(none)]> select count(*),kd_prov from db_tpm.tt_tpm group by kd_prov;
+----------+---------+
| count(*) | kd_prov |
+----------+---------+
|     1227 | 0       |
|     8807 | 11      |
|     7387 | 12      |
|     8297 | 13      |
|     9086 | 14      |
|     5484 | 15      |
|     3993 | 16      |
|     3310 | 17      |
|     8941 | 18      |
|     3747 | 19      |
|     5691 | 21      |
|    13173 | 31      |
|    31188 | 32      |
|    26660 | 33      |
|     4205 | 34      |
|    23027 | 35      |
|     7940 | 36      |
|     5484 | 51      |
|     4766 | 52      |
|     2485 | 53      |
|     5733 | 61      |
|     4452 | 62      |
|     8603 | 63      |
|    12022 | 64      |
|     3296 | 65      |
|     2555 | 71      |
|     5300 | 72      |
|    14018 | 73      |
|     5106 | 74      |
|     2233 | 75      |
|     1786 | 76      |
|     1304 | 81      |
|      882 | 82      |
|     1162 | 91      |
|     2595 | 94      |
+----------+---------+
35 rows in set (0.198 sec)
```

Figure 12: Query from spider server



```
MariaDB [(none)]> select count(*),kd_prov from tpm_db.tt_tpm group by kd_prov;
+----------+---------+
| count(*) | kd_prov |
+----------+---------+
|     1227 | 0       |
|     8807 | 11      |
|     7387 | 12      |
|     8297 | 13      |
|     9086 | 14      |
|     5484 | 15      |
|     3993 | 16      |
|     3310 | 17      |
|     8941 | 18      |
|     3747 | 19      |
|     5691 | 21      |
|    13173 | 31      |
|    31188 | 32      |
|    26660 | 33      |
|     4205 | 34      |
|    23027 | 35      |
|     7940 | 36      |
|     5484 | 51      |
|     4766 | 52      |
|     2485 | 53      |
|     5733 | 61      |
|     4452 | 62      |
|     8603 | 63      |
|    12022 | 64      |
|     3296 | 65      |
|     2555 | 71      |
|     5300 | 72      |
|    14018 | 73      |
|     5106 | 74      |
|     2233 | 75      |
|     1786 | 76      |
|     1304 | 81      |
|      882 | 82      |
|     1162 | 91      |
|     2595 | 94      |
+----------+---------+
35 rows in set (0.116 sec)
```

Figure 13: Query from the single database server (existing)



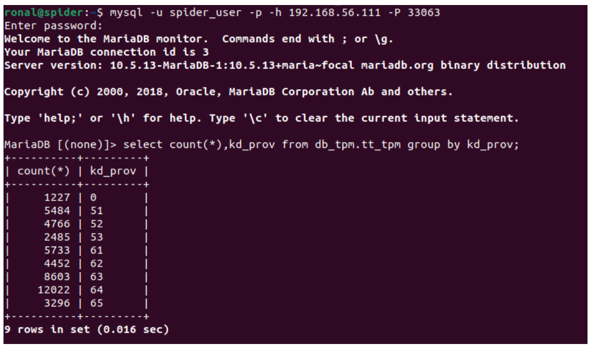
```
ronal@spider:~$ mysql -u spider_user -p -h 192.168.56.111 -P 33062
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.5.13-MariaDB-1:10.5.13+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> select count(*),kd_prov from db_tpm.tt_tpm group by kd_prov;
+----------+---------+
| count(*) | kd_prov |
+----------+---------+
|     8807 | 11      |
|     7387 | 12      |
|     8297 | 13      |
|     9086 | 14      |
|     5484 | 15      |
|     3993 | 16      |
|     3310 | 17      |
|     8941 | 18      |
|     3747 | 19      |
|     5691 | 21      |
|    13173 | 31      |
|    31188 | 32      |
|    26660 | 33      |
|     4205 | 34      |
|    23027 | 35      |
|     7940 | 36      |
+----------+---------+
16 rows in set (0.041 sec)
```

Figure 14: Query from the web node server



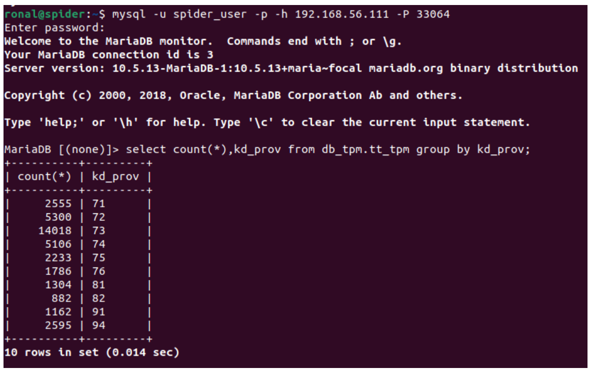
```
ronal@spider:~$ mysql -u spider_user -p -h 192.168.56.111 -P 33063
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.5.13-MariaDB-1:10.5.13+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> select count(*),kd_prov from db_tpm.tt_tpm group by kd_prov;
+----------+---------+
| count(*) | kd_prov |
+----------+---------+
|     1227 | 0       |
|     5484 | 51      |
|     4766 | 52      |
|     2485 | 53      |
|     5733 | 61      |
|     4452 | 62      |
|     8603 | 63      |
|    12022 | 64      |
|     3296 | 65      |
+----------+---------+
9 rows in set (0.016 sec)
```

Figure 15: Query from the node server



```
ronal@spider:~$ mysql -u spider_user -p -h 192.168.56.111 -P 33064
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.5.13-MariaDB-1:10.5.13+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> select count(*),kd_prov from db_tpm.tt_tpm group by kd_prov;
+----------+---------+
| count(*) | kd_prov |
+----------+---------+
|     2555 | 71      |
|     5300 | 72      |
|    14018 | 73      |
|     5106 | 74      |
|     2233 | 75      |
|     1786 | 76      |
|     1304 | 81      |
|      882 | 82      |
|     1162 | 91      |
|     2595 | 94      |
+----------+---------+
10 rows in set (0.014 sec)
```

Figure 16: Query from the node server

4. Conclusions

In this study, several things can be concluded:

1. In some journals about Sharding, sharding is more widely implemented in NoSQL DBMS such as MongoDB or InfluxdB. This study proves that sharding can also be done in relational DBMS such as MariaDB and MySQL.
2. The implementation of a distributed database can be done following the steps in the discussion above.
3. Each server can communicate with each other using a computer network and the spider server can perform data manipulation using defined users.
4. Based on the test results, the performance of the time query response is still better single database, according to this author, it's because :
    - Spider Server performs remote to server nodes causing I/O Network and storage that takes longer (response time) than single database server where I/O Network and Storage reside on the server itself.
    - In addition, when querying join table, spider server performs remote query execution to server nodes and queries on its local; this is thought to be because query latency increases from various two-way data traffic in the network.
    - Looking at information from the MySQL benchmark, the number of rows used data is more than 1 million - 10 million rows, while the data tested in the study amounted to 250,000. We think the use of data used is more minor.

5. Suggestions

1. Researchers need to optimize queries or tuning in the distributed database, especially using spider storage engines, by learning the features and advanced options.
2. Perform sharding by dividing the sharding more multiple nodes
3. Design and analyze the addition of proven code fields for supporting tables, such as tt_sertifikat
4. An analysis of the attributes is required, including partitions on each server node.
5. Conduct further research on distributed databases with sharding and replication methods that run High Availability (HA) and Fail-Over features using spider storage.
6. Perform data related to distributed databases using other methods such as using MySQL NDB, Max Scale and ProxySQL

References

[1] S. Bagui and L. T. Nguyen, "Database Sharding," *Int. J. Cloud Appl. Comput.*, vol. 5, no. 2, pp. 36–52, 2015, doi: 10.4018/ijcac.2015040103.

[2] Mulyani S, "Sistem Informasi Aplikasi Penggajian Karyawan Berbasis Web Pada Pt Panca Cipta Abadi," *Phys. Rev. E*, no. 1993, p. 24, 2020, [Online]. Available: http://ridum.umanizales.edu.co:8080/jspui/bitstream/6789/377/4/Muñoz_Zapata_Adriana_Patricia_Artículo_2011.pdf.

[3] Y. Suryansyah, "Evaluation of Hygiene and Sanitation Catering in Gayungsari Surabaya Street," *J. Kesehat. Lingkung.*, vol. 10, no. 2, p. 165, 2018, doi: 10.20473/jkl.v10i2.2018.165-174.

[4] J. Baydaoui, *Datenbanken und SQL*. 2013.

[5] Hendra and A. Widyastuti, "Studi Komparasi Menyimpan Dan Menampilkan Data Histori Antara Database Terstruktur Mariadb Dan Database Tidak Terstruktur Influxdb," *J. Teknol. Technoscientia*, vol. 12, no. 2, pp. 168–174, 2020.

[6] F. Castro-Medina *et al.*, "A New Method of Dynamic Horizontal Fragmentation for Multimedia Databases Contemplating Content-Based Queries," *Electron.*, vol. 11, no. 2, 2022, doi: 10.3390/electronics11020288.

[7] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *Prosisko*, vol. 7, no. 2, pp. 165–175, 2020, [Online]. Available: https://e-jurnal.lppmunsera.org/index.php/PROSISKO/article/view/2520/.

[8] E. Darwis, "Implementasi Basis Data Terdistribusi Menggunakan Mysql Pada Pt Thamrin Brothers Palembang," *Univ. Bina Darma*, pp. 1–8, 2019.

[9] Y. Randa, "국회선진화법' 에 관한 보론No Title'," 입법학연구, vol. 제13집 1호, no. May, pp. 31–48, 2016.

[10] B. P. Santoso, A. Noertjahyana, and J. Andjarwirawan, "Implementasi Distributed Database Pada Learning Management System Menggunakan Platform Redhat Openshift," *J. Infra*, pp. 1–5, 2020, [Online]. Available: http://publication.petra.ac.id/index.php/teknik-informatika/article/view/10501.

[11] J. Grech, "Designing and Implementing a Distributed Database for a Small Multi-Outlet Business," 2009.

[12] F. City, F. City, and F. City, "making QUERIES not," vol. 3, no. 2, pp. 119–127, 2015.

[13] A. Y. Seydim, "an Overview of Distributed Database Management," vol. 4, no. 2, pp.

207–214, 1998.

[14] P. R. Bhuyar, A. D. Gawande, and A. B. Deshmukh, “Horizontal Fragmentation Technique in Distributed Database,” *Int. J. Sci. Res. Publ.*, vol. 2, no. 5, pp. 1–7, 2012.

[15] T. Connolly and C. Begg, *Pearson.Database.Systems.A.Practical.Approach.to.Design.Implementation.and.Management.6th.Global.Edition.1292061189.pdf*. 2014.

[16] R. R and D. A. P. Rajan*, “Efficient Horizontal Scaling of Databases using Data Sharding Technique,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 5, pp. 590–593, 2020, doi: 10.35940/ijitee.e2418.039520.

[17] “New features and enhancements of Spider Storage Engine for sharding MariaDB Corporation Kentoku SHIBA Agenda.”

[18] P. Mavro, *MariaDB High Performance*. 2014.