

Joonas Rynänen

# Relational Database Clustering with MariaDB

## Galera

Bachelor of Business  
Administration

Business Information  
Technology

Autumn 2019



**KAMK • University  
of Applied Sciences**

## **Abstract**

**Author(s):** Ryyänen Joonas

**Title of the Publication:** Relational Database Clustering with MariaDB Galera

**Degree Title:** Bachelor of Business Administration, Business Information Technology

**Keywords:** database, replication, mariadb

The objective of this Bachelor's thesis goal was to build a highly available relational database system that is geographically distributed and can automatically handle failures.

The theory part of this thesis describes relational databases, basics of database replication, and an overview of different replication techniques used by relational database systems.

The practical part of this thesis describes building a geographically distributed relational database system and practical considerations of such a process.

## **Abstrakti**

**Tekijä(t):** Rynänen Joonas

**Työn nimi:** Relaatitietokannan klusterointi MariaDB Galeralla

**Tutkintonimike:** Tradenomi, Tietojenkäsittely

**Asiasanat:** tietokanta, replikointi, mariadb

Tämän opinnäytetyön tavoitteena oli rakentaa korkean saatavuuden relaatiotietokanta järjestelmä, joka on maantieteellisesti hajautettu ja pystyy toimimaan, vaikka tietyissä osissa esiintyisi vikatilanteita.

Työn teoria osa antaa perustiedot relaatiotietokannoista, tietokantojen replikoinnista, ja miten erilaisia replikaatiotekniikoita käytetään relaatiotietokanta järjestelmissä.

Työn käytännön osuus kertoo maantieteellisesti hajautetun tietokanta järjestelmän rakentamisesta ja sen aikana syntyneistä havainnoista.

## Contents

1	Introduction .....	2
2	Relational Database Systems.....	3
3	Replication in Relational Database Systems .....	4
3.1	CAP Theorem .....	4
3.2	Replication models.....	5
3.2.1	Asynchronous single primary replication.....	5
3.2.2	Synchronous single primary replication.....	5
3.2.3	Multi-primary replication.....	6
4	Replication Techniques in Relational Database Systems .....	7
4.1	Statement logs .....	7
4.2	Write ahead logs .....	7
4.3	Shared storage .....	7
4.4	Challenges.....	8
5	Building a cluster.....	9
5.1	Setup.....	10
5.1.1	Configuration.....	10
5.2	Replication .....	12
5.3	Monitoring.....	12
6	Conclusion .....	14
	Sources .....	15

## 1 Introduction

The goal of this Bachelor's thesis is to build a highly available relational database cluster that is geographically distributed to multiple datacenters. High availability means that the system is fault tolerant. This is done by running multiple database servers in a cluster and replicating data between them.

To achieve this goal, it is required to know about the basics of relational database systems. They are one of the most used database types in the world and trusted for business-critical data for their data integrity. Database replication, in simplified terms, means copying data between database servers. This is used to increase system availability or to scale them to handle higher loads. Replication is done either synchronously or asynchronously, and a solution is selected based on objectives that need to be archived. The most common technical solutions for replication are log or shared storage solutions.

To demonstrate real world applications for database replication a geographically distributed cluster for a web application is built. It is not meant to be production ready, so some required features are not implemented as a cost saving method.

## 2 Relational Database Systems

Databases are organized collections of data. They are managed by Database management systems that consist of a database engine that is responsible for creating, reading, updating, and deleting data in databases, schema model that is used for storing data and ensures data integrity, provides logging, auditing and other security features, networking, and interfaces for programming and users. Relational databases are based on a theory developed by E.F. Codd in the 1960s. Data in relational databases is stored in series of relations and identified by unique keys. Consistency in the databases is done with constraints.[1.][2.][3.]

MySQL is one of the world's most used relational database management systems. It is currently developed by Oracle, but it is open source and licensed under GNU General Public License. Originally it was developed to handle larger databases than alternatives and utilize CPU, RAM, and IO resources more efficiently. Oracle provides commercial support with MySQL Enterprise Edition, which also includes more advanced features and management tools. It is also available as a cloud service from Oracle Cloud. Customers include Google, Facebook, and Netflix. MySQL has been criticized for only implementing support for old structured query language standard. However, with a recent release of version 8.0 newer version of the standard is implemented.[4.][5.][6.]

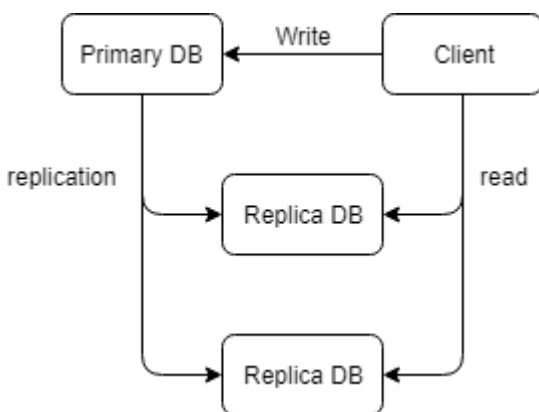
In a ranking done by db-engines.com (February 2019) at top ten there were seven relational database systems (see table 1). The most popular ones were Oracle, MySQL (Open Source system developed by Oracle), and Microsoft SQL Server.[7.]

Rank	System	Model
1.	Oracle	Relational
2.	MySQL	Relational
3.	Microsoft SQL Server	Relational
4.	PostgreSQL	Relational
5.	MongoDB	Document
6.	IBM DB2	Relational
7.	Redis	Key-Value
8.	Elasticsearch	Search
9.	Microsoft Access	Relational
10.	SQLite	Relational

Table 1: Most popular database systems in February 2019 ranked by db-engines.com

### 3 Replication in Relational Database Systems

Replication means actively synchronizing changes between database servers, leading them to contain the same data. It is a required feature in modern database systems and is often utilized to increase system availability, by decreasing recovery time in a case of failure. Combined with query routing in software, a proxy, or client-side load balancing solution it can be used to scale database systems to handle more load (see picture 1). The most used replication model with relational databases is the single primary replication where all writes go to one specific node and are replicated from it to others. The node that handles writes is known as primary and the others as replicas. MySQL uses a statement-based log replication technique and has support for both asynchronous and synchronous single primary replication models.[3.]



Picture 1: Database system scaled to multiple servers

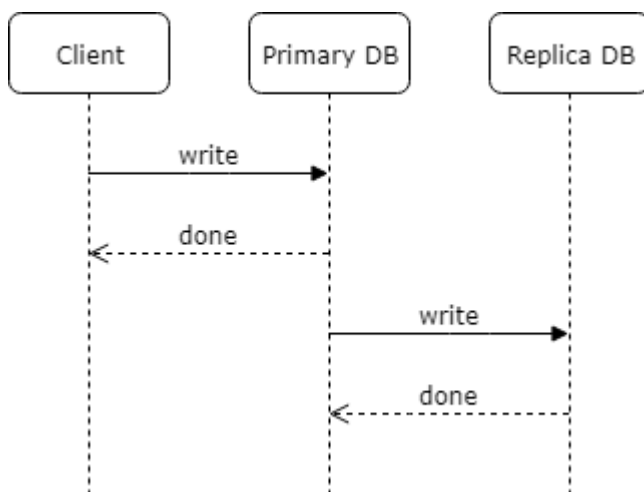
#### 3.1 CAP Theorem

CAP theorem was published by Eric Brewer in 2000. It states in that distributed systems you can have two of the following features: consistency, availability, or partition tolerance. Consistency means that data is same between all servers. Availability means that all operations always succeed. Partition tolerance means that the system will work when servers can't communicate. In most cases the trade will be done between consistency and availability. In consistent systems, data that is read from multiple servers at the same time will return the same answer. In available systems, reads and writes always succeed even if replication to other servers is not possible. This can lead to inconsistencies between the servers when writing or that one might not receive most recent data when reading.[8.]

## 3.2 Replication models

### 3.2.1 Asynchronous single primary replication

Asynchronous replication, sometimes shortened as async replication, is done by writing to primary database and to replica database in a different process (see picture 2). This means that there is no extra latency when writing compared to no replication, but it can lead inconsistencies between primary and replica databases. If the primary database would fail that could lead to permanent data loss.[3.]

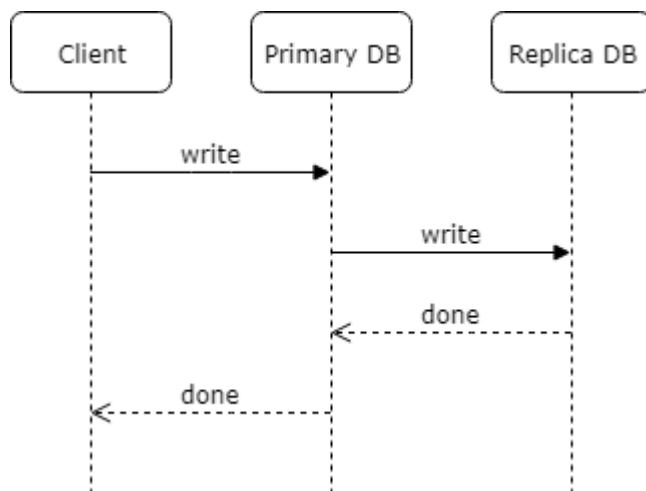


Picture 2: Asynchronous replication process

### 3.2.2 Synchronous single primary replication

Synchronous replication, sometimes shortened to sync replication, is done by writing to primary and replica databases in one process (see picture 3). This means that writing will take longer compared to system a with no or asynchronous replication. However, it guarantees consistency between primary and replica databases, and no data would be lost if the primary database would fail.[3.]





Picture 3: Synchronous replication process

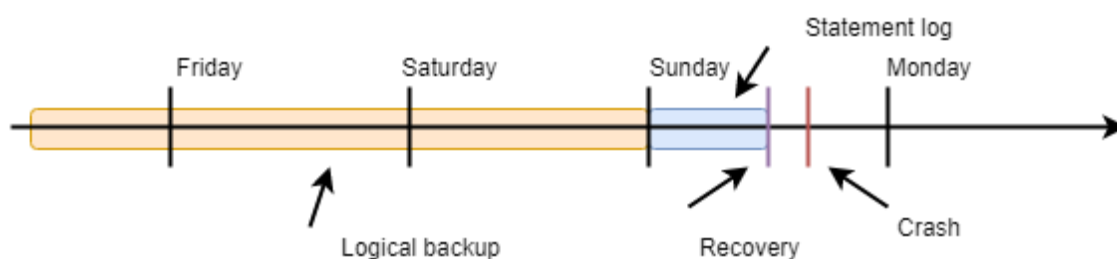
### 3.2.3 Multi-primary replication

In multi-primary replication model, all nodes in the system can accept write statements, which are then synchronously replicated to other nodes. This model can be used to increase system availability, but it introduces a performance hit to write statements and requires more technical knowledge to architecture and administrate. Multi-primary replication usually requires a quorum between primary nodes to accept write statements, but other techniques are also available.[3.]

## 4 Replication Techniques in Relational Database Systems

### 4.1 Statement logs

In statement log-based replication all write statements are written to a log, the log is transported to other servers, and the write statements are executed there. It doesn't require much network resources and is often used to replicate between different datacenters. But some statements, for example those using random function, don't work with it. Statement log-based replication is used by MySQL. Statement logs can be used with logical backups to restore databases to a specific point in time (see picture 4). [3.]



Picture 4: Combining data from a logical backup and statement log to recover database to specific point in time

### 4.2 Write ahead logs

In write ahead log based replication data of write statements is written to a log. The log is shipped to other servers. Data is then applied from the log to disks. This is faster than statement log-based replication which consumes more network resources. Write ahead log based replication is used by PostgreSQL.[3.]

### 4.3 Shared storage

In shared storage replication multiple database servers share the same data storage. It can be a block device or a software designed storage solution. Block level replication is always synchronous. However only one of the database servers can be active at the time. This limits the

shared storage-based replication only for high availability systems. Microsoft SQL Server Failover Cluster uses shared storage-based replication solution.[3.]

#### 4.4 Challenges

Replication increases complexity of database systems and with it come some challenges. With large databases, the time for building new database replicas increases and this increases the time required to recover after failure. Building a new replica can also increase the system load on primary servers. When using asynchronous replication there might be some delay on the replication. This could lead to loss of data in a case of failure. In a case of primary database systems failing, one of the primary servers needs to be promoted to the primary database server role. This failover process can be automated but can cause some issue.[3.]

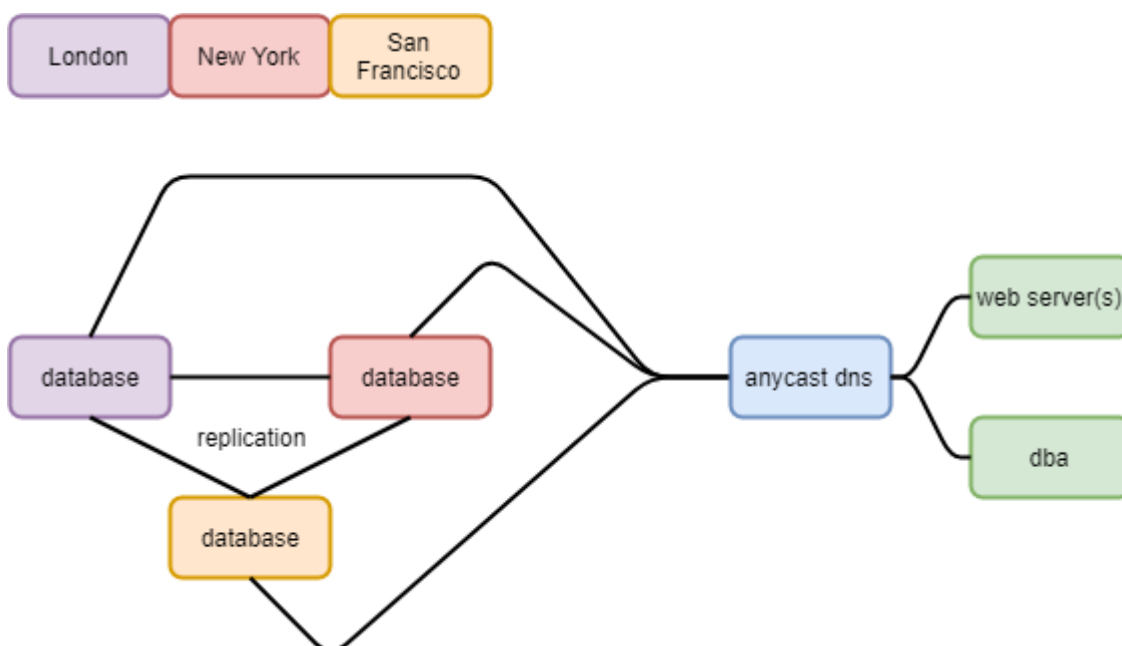
Automatic failover process can lead to non-operational system or cause data loss. This happened to GitHub in October 2018. When network partition between datacenters lead to automatic re configuration of complex database replication setup.[9.]

## 5 Building a cluster

For the practical part of this thesis, a highly available database system was built using MariaDB and Galera Cluster.

MariaDB is one of the world's most used relational database systems, it is used in everything from a simple website to a critical banking system. It was developed by original MySQL developers as drop-in replacement after Oracle purchased it with Sun Microsystems. Galera Cluster is multi primary replication solution for MariaDB and MySQL. Galera can be used to scale read intensive systems or increase system availability by replicating databases.[10.][11.]

The system uses Galera's multi-primary replication technique and is geographically distributed to three datacenters in London, New York, and San Francisco (see picture 5). All these sites are completely similar, and the design of the system allows anyone of these locations to become unavailable and the services to be provided from the remaining online. This system is not designed to be production ready and some non-database related features that would be necessary on a production system weren't implemented as a cost saving method. These included anycast dns solution for routing traffic to the system, health check backend required for the anycast dns, and backups. The system was tested by randomly disconnecting a node from the cluster and it remained operational, fulfilling the requirements for the highly available system.



Picture 5: Architecture of the system

## 5.1 Setup

Instances for the system were deployed from DigitalOceans web panel to datacenters in New York, San Francisco, and London using latest CentOS 7 image. To the instances MariaDB version 10.4 was installed using their official repository, which makes upgrading to newer versions easier compared to installing all required packages by hand. Database connections between the instances are encrypted. Inbound and Outbound connections are filtered with a firewall and access to ports used by MariaDB is limited to instances in the system (see picture 6).

Type	Protocol	Port Range	Destinations
ICMP	ICMP		All IPv4 All IPv6
HTTP	TCP	80	All IPv4 All IPv6
HTTPS	TCP	443	All IPv4 All IPv6
MySQL	TCP	3306	nyc sfo ldn
Custom	TCP	4567	nyc sfo ldn
Custom	TCP	4568	nyc sfo ldn

Picture 6: Firewall rules for outbound connections

### 5.1.1 Configuration

MariaDB configuration is specific to galera replication(see picture 7). The location of the configuration file varies between operating systems.

```
[galera]
# Enable Galera write set replication
wsrep_on = ON

# Address of cluster nodes to which connect during startup
# Best practice is to list all cluster nodes
# If leaved empty new cluster will be bootstrapped
wsrep_cluster_address = gcomm://

# Location of write set replication library
```

```

# On Ubuntu default location is /usr/lib/libgalera_smm.so
# On Red Hat default location is /usr/lib64/libgalera_smm.so
wsrep_provider = /usr/lib64/libgalera_smm.so

# List of options passed to write set replication provider
# This is used to fine tune replication
# And configure ssl encryption for connections
wsrep_provider_options = "
socket.ssl_ca=/etc/my.cnf.d/certificates/ca.pem;
socket.ssl_cert=/etc/my.cnf.d/certificates/server-cert.pem;
socket.ssl_key=/etc/my.cnf.d/certificates/server-key.pem"

# Method used to transfer database state snapshots in the cluster
# Can be utilized in backing up the cluster
wsrep_sst_method = mysqldump

# Authentication for selected write set replication method
wsrep_sst_auth = <user>:<password>

# Online Schema Upgrade Method
# TOI = total order isolation
# Data definition language is processed globally in the cluster
# Guarantees data consistency, but causes databases to be locked
# RSU = Rolling Schema Upgrade
# Data definition language is processed in local node
# Requires running schema changes separately at all nodes
wsrep_OSU_method = TOI

# Number of threads used to apply write sets
wsrep_slave_threads = 1

# Command that is called each time their changes to cluster mem-
bership
# Useful for monitoring and dynamic configuration of a cluster
wsrep_notify_cmd = /usr/bin/wsrep_notify

# Replication format
# Galera requires row as a format
binlog_format = row

# Lock mode for generating auto increment values
# Galera requires consecutive lock mode
innodb_autoinc_lock_mode = 2

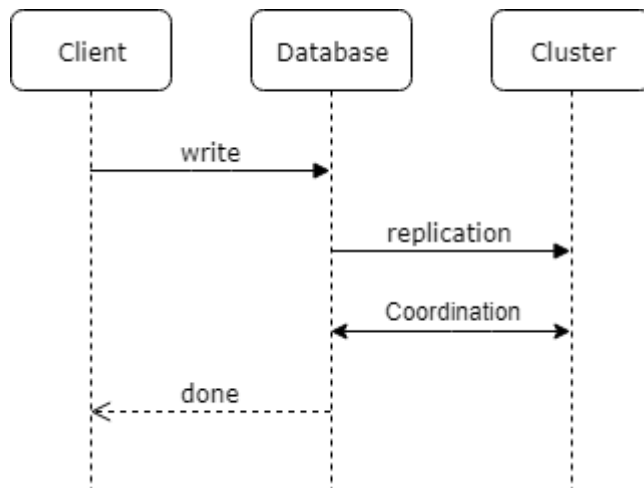
# Flush InnoDB redo log to disk once a second, not on commit
# Used to increase performance in Galera system
# Fault tolerance and consistency is handled on cluster not local
level
innodb_flush_log_at_trx_commit = 0

```

Picture 7: MariaDB Configuration

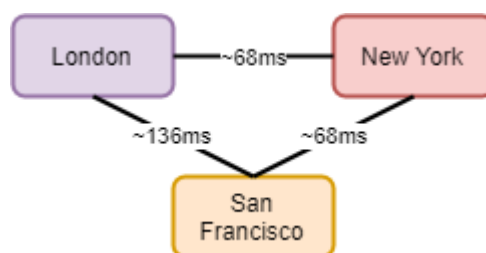
## 5.2 Replication

Galera's replication technique is very similar to a synchronous single primary replication, but extra steps are added to the process to coordinate write statement execution between nodes (see picture 8). This enables ordering write statements globally across the cluster, enabling all the nodes in the cluster accepting them. Replication connections are encrypted with TLS versions 1.2 or 1.3 and nodes are authenticated when connecting to each other. [12]



Picture 8: Galera replication technique

As the replication is done synchronously and has required coordination between nodes when executing write statements, there is more latency than in the non-clustered system. It amounts roughly to double of network round trip time between server with longest round trip. In the case longest latency was between nodes in London and San Francisco (See picture 9).



Picture 9: Latencies between datacenters

## 5.3 Monitoring

DigitalOcean provides free monitoring services for servers on its compute platform. It works by collecting data with an agent installed to the server and sending the data to DigitalOcean's service.

This data is available in graph to inspect the server's performance metrics (see picture 10) and can be used to create monitoring alert policies.



Picture 10: Graph of performance data

The alert policies can send notifications to an email box or Slack channel when they are triggered (see picture 11) and when the issue is resolved by metrics decreasing below the alert policy threshold.

---

 DigitalOcean <support@support.digitalocean.com>  
Sat 2019-07-13 12:38  
You 

Memory Utilization is currently at 37.34%, above threshold of 30.00% for more than 5 minutes.

View droplet nyc: <https://cloud.digitalocean.com/droplets/149570371?i=9791c8>  
IP: 67.205.133.150

Picture 11: Notification from monitoring alert policy



## 6 Conclusion

To develop the system for production usage some changes should be made. To automate configuration changes and ease administration of the system configuration management and automation tool like ansible or puppet should be deployed. Ansible supports automating instance deployment on DigitalOcean making it easier to expand the system if needed. MariaDB supports encrypting data-at-rest, so it should be used to prevent access to data contained in databases via vulnerability in the server or storage platforms. Backups should be made either of the servers to DigitalOcean's backup service or logically from databases on the system. DigitalOcean has object storage service which could be utilized to store logical backups. Security-Enhanced Linux should be used to enhance systems security and suitable audit policies should be developed.

Relational databases have been used for almost fifty years and are still the most used type of database. Their availability or performance can be increased with replication but it increases the level of knowledge required to architecture and administrate these systems.

## Sources

1. Hugh Darwen. An Introduction to Relational Database Theory. Ventus Publishing ApS. 2012
2. What is database management system (DBMS)? Referenced 24.12.2018. Available from <https://searchsqlserver.techtarget.com/definition/database-management-system>
3. Laine Campell, Charity Majors. Database Reliability Engineering. O'Reilly. 2017
4. What is MySQL? Referenced 21.01.2019, available from <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>
5. One Giant Leap For SQL: MySQL 8.0 Released. Referenced 21.01.2019. Available from <https://modern-sql.com/blog/2018-04/mysql-8.0>
6. MySQL. Referenced 04.02.2019. Available from <https://www.mysql.com/>
7. DB-Engines Ranking. Referenced 04.02.2019. Available from <https://db-engines.com/en/ranking>
8. A Primer on Database Replication. Referenced 08.01.2019. Available from <https://www.briantorti.com/replication/>
9. October 21 post-incident analysis. Referenced 28.01.2019. Available from <https://blog.github.com/2018-10-30-oct21-post-incident-analysis/>
10. About MariaDB. Referenced 28.01.2019. Available from <https://mariadb.org/about/>
11. What is MariaDB Galera Cluster? Referenced 12.08.2019. Available from <https://mariadb.com/kb/en/library/what-is-mariadb-galera-cluster/>
12. Certification-Based Replication. Referenced 18.08.2019. Available from <https://galeracluster.com/library/documentation/certification-based-replication.html>