Distributed Database System

Faculty of Computer Science
Assiut University

Team Members:

Aya AbdelLatif Mohammad
Safaa Mohammad Shahin
Shahd Atta Elkareem Taha
Marwa Amer Morsi

May 2025

1 System Overview

This report details the design and implementation of a distributed database system using the **Go programming language** and a relational **MYSQL database engine**. The objective of this project was to implement core distributed systems concepts such as **replication**, and **inter-node communication**, all while supporting essential database operations like creation, queries, and updates.

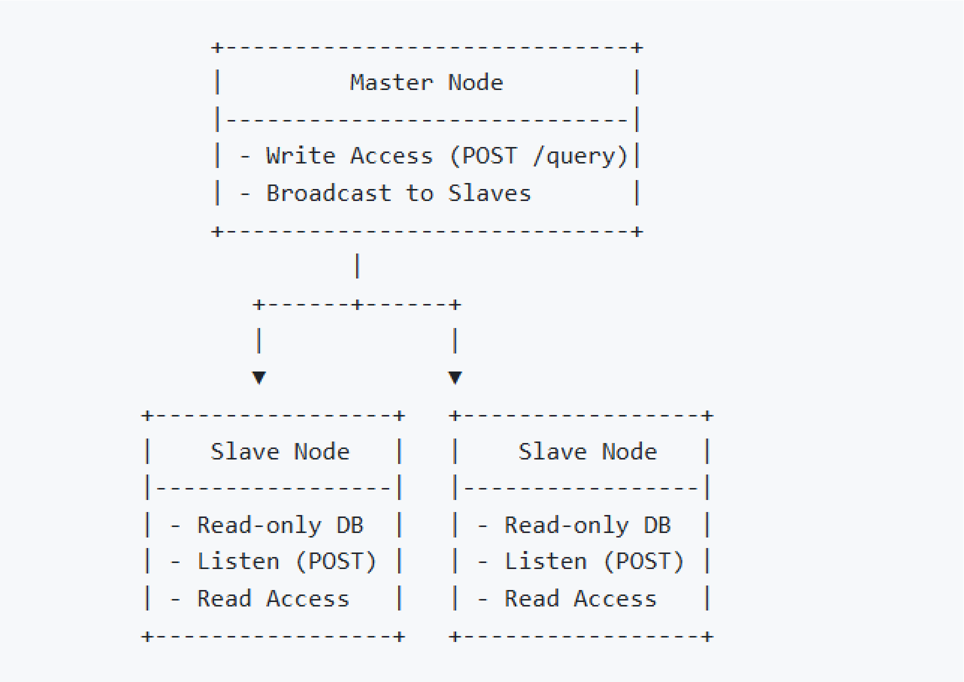
- **1 Master Node**: Handles all write operations and query replication.
- **Multiple Slave Nodes**: Handle read operations and maintain synchronized data copies.

Key Features

- HTTP-based communication between nodes.
- Automatic query replication from master to slaves.
- Support for standard MySQL operations (CRUD).
- Graphical User Interface (GUI):

2 System Architecture

The system follows a star topology with the master at the center:



```
     +-----------------------------+
     |         Master Node         |
     |-----------------------------|
     | - Write Access (POST /query)|
     | - Broadcast to Slaves       |
     +-----------------------------+
               |
        +------+------+
        |             |
        ▼             ▼
+-----------------+   +-----------------+
|    Slave Node   |   |    Slave Node   |
|-----------------|   |-----------------|
| - Read-only DB  |   | - Read-only DB  |
| - Listen (POST) |   | - Listen (POST) |
| - Read Access   |   | - Read Access   |
+-----------------+   +-----------------+
```

Figure 1: System Architecture Diagram

Component Roles

Component	**Responsibilities**
Master Node	• Database/table creation/dropping • Query execution and replication • System coordination
Slave Nodes	• Read-only query execution • Replicated write application • Local data synchronization

3 Implementation Details

Technical Stack

- **Programming Language**: Go (Golang)
- **Database Engine**: MySQL
- **Communication Protocol**: HTTP/JSON

4 Operation Examples

Database Operations

```
# Create database
curl -X POST http://localhost:8080/query -H
"Content-Type: application/json" -d
"{\"database\": \"\", \"query\": \"CREATE DATABASE Company\"}"

# Create table
curl -X POST http://localhost:8080/query -H
"Content-Type: application/json" -d "{\"database\":\"Company\", \"query\":\"
CREATE TABLE Employee (id INT PRIMARY KEY, name VARCHAR(100))\"}"
```

Data Operations

```
# Insert data (replicated to slaves)
curl -X POST http://localhost:8080/query \
     -d '{"database": "Company",
          "query": "INSERT INTO Employee VALUES (1, 'John')"}'

# Query data (executed on slaves)
curl -X POST http://localhost:8081/query \
     -d '{"database": "Company",
          "query": "SELECT * FROM Employee"}'
```

5 Challenges and Solutions

- **Query Parsing**: Implemented robust mySQL parsing for DROP operations.
- **Replication Consistency**: Used Base64 encoding for query transmission.
- **Network Configuration**: Established local network communication protocols.
- **Concurrency Control**: Implemented Go routines for parallel slave updates.

6 Conclusion

The implemented system successfully demonstrates core distributed database concepts including:

- Data replication and synchronization -based access control.

Future enhancements could include:

- Transaction support.