

DATABASES

1. Purpose of the database

The user interface would allow the users to manipulate the board configuration, and to access

tutorials for learning about RF communication. The web server connects to the user interface on

the client side, and communicates with the user and board database. The user database

contains all user information, for both students and instructors.

The information that needs to be included in the database comprises of user data and board data. As

previously mentioned, user data contains values the user needs to authenticate to the application,

while as board data describes the various configurations for the board.

A solution has been derived where board data configurations are stored in a database table, similar to

the user data, as seen on Figure 24. In user data, the ‘user\_type’ field stores whether the user is a

student or an instructor, and the ‘board\_data’ is a string that stores the user’s last board

configuration. However, according to industry standards, storing configuration information field by

field in a database is very inefficient. Therefore, configuration data are most often stored in JSON files,

as shown in Figure 25. Since the board data is organized based on the various subsystems, JSON

makes it very easy to classify the fields. The tradeoff is that the entire JSON objects needs to be

rewritten after each modification.

3.4 Database Design

3.4.1 Problem and Purpose

StreamingOS requires a way to store data related to every user which could range from credentials required to authenticate a user into the system to storing information regarding virtual machine been assigned to each user and much more. Consequently, a database becomes an essential component of this subsystem system. The following paragraphs would deep dive into an analysis describing the design decisions made in order to formulate the database design best suited for this subsystem.

3.4.2 SQL (RDBMS) vs NoSQL

In the database and data warehouse world it comes down to two types structure for designing a database available for the developers to store data, either SQL (Structured Query language) an RDBMS (Relational Database Management System) or NoSQL, a non-relational database system. The differences between the two options are rooted in the way they are designed, types of data each can support, and the way each type stores and represents the data.

A relational database is modelled around relationally structured entities, which represents an object in the real-world; for example, a person or an online store catalog whereas a non-relational database (NoSQL) is a document-structured and distributed, holding information in a folder-like Hierarchy storing data in an unstructured format[1]. There are many variations of NoSQL databases ranging from key-value pairs to object-oriented designs, or even graph data structures for implementing the database [2]. The two types are in short known as SQL for relational database system and NoSQL for non-relational database system, which reflects that NoSQL databases are not written in SQL only [1].Relations is regarded as sets in mathematics, each containing certain attributes collectively representing the information or data in SQL terms. SQL is a special purpose programming language for RDBMS databases, is used to access and modify RDBMS data. Since this is a traditional system, which has already been widely adopted by industry, there exist a wide array of SQL servers that implement these principles and allow for easy usage for most users who wish to use a RDBMS database [3].

In terms of flexibility and schema an RDBMS uses a fixed schema, which means information cannot be stored into a database and then changed to contain new information which means the columns must be decided and locked before any data entry and each row must contain data for each column unless specified to handle null values. It can be amended but any kind of change would affect the whole database and can be only done going offline. A NoSQL database uses a dynamic schema, which would allow the information to be added whenever required and any extra information can be stored and not even comply with the existing schema [2]. In terms of scalability, an RDBMS (SQL database) scales vertically which means that more data leads to a bigger server and storing data across multiple severs would reduce the speed for any kind of data related operation. The NoSQL database expands or scales horizontally across multiple servers. This ability of horizontal scaling by NoSQL enables it to handle more data by simply adding new servers to increase its load capacity, whereas RDBMS databases scale vertically, requiring an increase in resources such as CPU or RAM on existing servers to handle more load. NoSQL is generally faster for simple queries, whereas RDBMS is generally faster and more robust for more complex queries. One last feature to consider is the A.C.I.D compliancy where the complete form of the property is Atomicity, Consistency, Isolation and Durability. NoSQL advantage in term of speed and scalability comes at the cost losing out in the A.C.I.D compliancy while the RDBMS is A.C.I.D compliant [2][3].

After comparing the two types of the databases designs the RDBMS model was chosen for this project due to its robustness and reliability which is the top priority. In order to reach the spec requirement of being able to authenticate users and having access to structured data about each user and virtual machine assigned to each one of them and being able to support multithreaded access (which is required if 10 users are required to be served per unique device per minute) it is necessary to have a reliable system that implements ACID properties and is able to serve large queries. The queries from the API will mostly be complex and compound, requiring many joins across many different blocks of data to retrieve all the required information to serve task or display analytics. Thus, the pros of using RDBMS is worth more when compared to pros of using the NoSQL database model.

**3.4.3 MySQL vs Postgres**

There are a lot of options in terms of systems using the RDBMS model. The top two most promising options includes MySQL and PostgreSQL. MySQL is considered as the most popular open-source RDBMS based on the DB-Engine ranking since 2012. This open-source system contains many

features and with the help of its popularity contains of lot of resources, documentation and third-party integration tools that support it. The main reason behind MySQL’s popularity is the speed, security and the reliability it provides and all of this at the expense of full adherence to standard SQL. A MySQL database can be accessed through multiple separate daemon processes because a server process stands between the other applications and the database itself for greater control over for the one which has access to the database.It is designed in way that usually allows executed queries to be placed in functions known as stored procedure and executed repeatedly. However, it does have some limitations because in terms of functionality with the omission of certain SQL standards, and it may not be as

reliable as other RDBMS databases when it comes to certain advanced features such as

references, auditing, and transactions [4].

While on the other side PostgreSQL is a well-known largescale open source RDBMS which aims to adopt most of the SQL standards when compared to the MySQL. It is different from other RDBMS in that it implements advanced technology to keep it A.C.I.D. compliant while supporting concurrency where it handles multiple tasks at the same time without any loss in performance. It achieves it by not avoiding any read locks and is implemented using multi version concurrency control (MVCC) instead, which also maintains the A.C.I.D. compliance. Similar to MySQL, PostgreSQL also supports functionality of stored procedures. Even both MySQL and PostgreSQL offer the trigger functionality which enables certain stored procedures to run when a certain condition is met, PostgreSQL provides additional advanced trigger features not supported by MySQL. One of the issues is that MySQL is faster and takes less memory when compared to PostgreSQL. For every new client connection, PostgreSQL forks a new process where each new process is allocated about 10MB of memory, which adds up quickly for databases with many concurrent connections. It is not as popular as MySQL and as thus doesn’t have a lot of documentation for new developers [4].

Based on the information above as a base to choose the system, MySQL presents as the

perfect candidate for the project because speed and high performance are imperative for the project.

It also ensures reliability and correctness, which are critical technical specifications and form the foundation from used for keeping record of users and virtual machine assigned to each user.

The stored procedures also help automated scripts to run easily and systematically.

**3.4.4 Database Design and Schema**

Following the above analysis, the final database design is implemented using RDBMS model.

The language used for this model is Postgres SQL Database and have decide to go ahead with MyS with data being stored and represented in tables. Catering to the needs

of the product, the schema design is shown in Figure 3.

[1] - <https://blog.panoply.io/sql-or-nosql-that-is-the-question>

[2] - https://dataconomy.com/2014/07/sql-vs-nosql-need-know/

[3] - "SQL", Wikipedia, 2019. [Online]. Available: https://en.wikipedia.org/wiki/SQL. [Accessed: 23 Jun 2019].

[4] - [18] O. Tezer, "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management

Systems | DigitalOcean", Digitalocean.com , 2016. [Online]. Available:

https://www.digitalocean.com/community/tutorials/sqlitevsmysqlvspostgresqlacomparisonofrelationald

atabasemanagementsystems.

[Accessed: 28Jun2016].