2016 IEEE International Conference on Advances in Computer Applications (ICACA)

MySQL and NoSQL database comparison for IoT application

Sharvari Rautmare - Dept. of E & TC Sinhgad College of Engineering, Pune, India. [sharvari318@gmail.com](mailto:sharvari318@gmail.com)

Dr. D. M. Bhalerao - Dept. of E & TC Sinhgad College of Engineering, Pune, India.

[dmbhalerao@sinhgad.edu](mailto:dmbhalerao@sinhgad.edu)

**Abstract— Internet of Things (IoT) concept has been around in tech world for few years now. IoT focuses on connection of number of smart devices. In near future, IoT will have applications in various domains and these applications are going to produce tremendous amount of data. With the continuous generation of heterogeneous data, problem arises to store, transfer & manage the data efficiently**. **Traditional database systems used Structured Query Language (SQL) database which has supported all the user requirements along with simplicity, robustness, flexibility, scalability, performance**. **But the main limitation they are facing is their static schema which is making RDBMS not suitable for IoT applications.** **On the other hand, NoSQL databases emerging in market have claimed to perform better than SQL database**. **The NoSQL databases are nonrelational, schema free, no joins, easy replication support, horizontally scalable, etc.** **Does NoSQL perform better than SQL in all application scenarios**? **An effort to answer the same has been made in this paper. This paper compares SQL and NoSQL databases for a small scale IoT application of water sprinkler system and investigates whether NoSQL performs better than SQL in different scenarios.**

**Khái niệm về Internet kết nối vạn vật (IoT) đã xuất hiện trong thế giới công nghệ vài năm nay IoT tập trung vào kết nối của số lượng thiết bị thông minh. Trong tương lai gần, IoT sẽ có các ứng dụng trong nhiều lĩnh vực khác nhau và các ứng dụng này sẽ tạo ra lượng dữ liệu khổng lồ. Với việc tạo ra dữ liệu không đồng nhất liên tục, phát sinh vấn đề để lưu trữ, chuyển và quản lý dữ liệu một cách hiệu quả**. **Các hệ thống cơ sở dữ liệu truyền thống đã sử dụng cơ sở dữ liệu ngôn ngữ truy vấn có cấu trúc (SQL), SQL đã hỗ trợ tất cả các yêu cầu của người dùng với sự đơn giản, mạnh mẽ, linh hoạt, khả năng mở rộng, hiệu suất**. **Nhưng hạn chế chính mà SQL phải đối mặt là lược đồ tĩnh của nó, điều này làm cho hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) không phù hợp với các ứng dụng IoT**. **Mặt khác, cơ sở dữ liệu NoQuery đang nổi lên trên thị trường và hoạt động tốt hơn cơ sở dữ liệu SQL.** **Các cơ sở dữ liệu NoSQL không có mối quan hệ, không có lược đồ, không có join, hỗ trợ nhân rộng dễ dàng, có thể mở rộng theo chiều ngang, v.**v. **Vậy NoQuery có hoạt động tốt hơn SQL trong tất cả các loại ứng dụng không**? **Bài báo này sẽ trả lời cho câu hỏi đó**. **Bài báo này so sánh các cơ sở dữ liệu SQL và NoSQL cho một ứng dụng IoT quy mô nhỏ của hệ thống phun nước và điều tra xem NoSQL có hoạt động tốt hơn SQL trong các tình huống khác nhau hay không.**

Keywords— Database Management System (DBMS); Internet of Things (IoT); NoSQL; SQL;

**Từ khóa - Hệ thống quản lý cơ sở dữ liệu (DBMS); Internet kết nối vạn vật (IoT); NoSQL; Ngôn ngữ truy vấn có cấu trúc (SQL);**

1. INTRODUCTION

**A lot of research work is going on in Internet of Things (IoT) domain. IoT refers to any object/thing around us which is connected to internet. This object is also able to communicate through internet [1].** **Sensors are attached to these objects/things, like temperature sensor connected to oven, fridge or just to measure a room temperature.** **IoT is network of such things which may communicate with each other or with user.** **IoT is having application in various domains like smart city, industrial, medical services, etc. As, the application area is wide, the type of data generated by these things is heterogeneous. In near future, data generated by these IoT applications will be big data which can be further analyzed to make the optimum use of resources.** **Also, the use of IoT applications will increase the generation of real-time data, making it challenging to store and manage the data**. **Several new technologies and database management systems (DBMS) are emerging to handle the data growth and to improve the system performance.** **SQL databases have been the most popular database systems so far, question arises whether the traditional relational databases will perform at par with ever increasing user requirements.**

1. Giới thiệu

**Rất nhiều công việc nghiên cứu đang diễn ra trong lĩnh vực Internet of Things (IoT). IoT đề cập đến bất kỳ đối tượng / sự vật xung quanh chúng ta được kết nối với internet. Đối tượng này cũng có thể giao tiếp qua internet [1].** **Các cảm biến được gắn vào các đối tượng / vật này, như cảm biến nhiệt độ được kết nối với lò nướng, tủ lạnh hoặc để đo nhiệt độ phòng. IoT là mạng lưới những vật có thể giao tiếp với nhau hoặc với người dùng.** **IoT đang có những ứng dụng trong các lĩnh vực khác nhau như thành phố thông minh, dịch vụ công nghiệp, dịch vụ y tế, v.v. Vì, khu vực ứng dụng rộng, loại dữ liệu được tạo bởi những ứng dụng này là không đồng nhất. Trong tương lai gần, dữ liệu được tạo bởi các ứng dụng IoT này sẽ là dữ liệu lớn (big data), nó có thể dùng để phân tích sự tối ưu cho các nguồn lực.** **Ngoài ra, việc sử dụng các ứng dụng IoT sẽ tăng việc sản sinh lượng dữ liệu thời gian thực, làm cho nó khó khăn để lưu trữ và quản lý dữ liệu.** **Một số công nghệ mới và hệ thống quản lý cơ sở dữ liệu (DBMS) đang nổi lên để xử lý sự tăng trưởng dữ liệu và cải thiện hiệu năng của hệ thống.** **Cơ sở dữ liệu SQL đã được dùng cho các hệ thống cơ sở dữ liệu phổ biến nhất từ xưa cho đến nay, câu hỏi đặt ra là liệu các cơ sở dữ liệu quan hệ truyền thống có đáp ứng được với yêu cầu ngày càng tăng của người dùng.**

**A new class of database is emerging and is referred to as NoSQL database [2]. NoSQL databases store data in a different manner as compared to the traditional methods of relational database systems. They are meant for data of schema-less structure.** **NoSQL databases can be easily distributed and hence they provide high scalability and availability. These are the properties which are actually required to manage big data or Internet of Things data.**

**Một lớp mới của cơ sở dữ liệu đang nổi lên và được gọi là cơ sở dữ liệu NoQuery [2]. Cơ sở dữ liệu NoSQL lưu trữ dữ liệu theo cách khác như so với các phương pháp truyền thống của cơ sở dữ liệu quan hệ hệ thống. NoSQL cho dữ liệu của cấu trúc không có lược đồ.** **Cơ sở dữ liệu NoSQL có thể được phân phối dễ dàng và do đó chúng cung cấp khả năng mở rộng cao và luôn sẵn có. Đây là những các thuộc tính thực sự cần thiết để quản lý dữ liệu lớn hoặc dữ liệu IoT.**

1. RELATED WORK

**To realize the Internet of Things technology, a lot research is going on in different fields related to IoT.** **IoT data is going to be heterogeneous as there are various applications of IoT in varied domains.** **The data generated by these IoT applications is further supposed to be processed and analyze so as to optimize the application further.** **So, tasks like Data collection, storage, data processing, generating reports are managed by the database management system.** **To handle the huge amount of data and make the data retrieval faster, authors [3] have proposed the design, implementation and evaluation of three data retrieval approaches for RFID Tags.** **Alternate, Sequential & parallel sensor sampling scheduling schemes proposed were compared based on parameters such as time delay, power consumption, communication range & read rate.** **Limited use of data and number of sensors is the limitation of the study.** **Authors [4] are focusing on application oriented work.** **Emergency medical services related IoT application is used to explore data accessing methods.** **For various IoT applications, question of whether SQL or NoSQL databases prove to be better, still remains unanswered.** **Such comparative study are done [5] [6] with data stored in cloud. Analysis of SQL and NoSQL database for specific IoT application is of our interest.**

**CÔNG VIỆC LIÊN QUAN**

**Để hiện thực hóa công nghệ Internet of Things, rất nhiều nghiên cứu đang diễn ra trong các lĩnh vực khác nhau liên quan đến IoT.** **Dữ liệu IoT sẽ không đồng nhất vì có nhiều ứng dụng khác nhau của IoT trong các lĩnh vực khác nhau. Dữ liệu được tạo bởi các ứng dụng IoT được cho là sẽ được xử lý và phân tích để tối ưu hóa ứng dụng hơn nữa. Vì vậy, các tác vụ như thu thập dữ liệu, lưu trữ, xử lý dữ liệu, tạo báo cáo được quản lý bởi hệ thống quản lý cơ sở dữ liệu.** **Để xử lý lượng dữ liệu khổng lồ và làm cho việc truy xuất dữ liệu nhanh hơn, các tác giả [3] đã đề xuất thiết kế, thực hiện và đánh giá trong ba cách tiếp cận truy xuất dữ liệu cho thẻ RFID.** **Đề án lấy mẫu cảm biến tuần tự & song song đề xuất được so sánh dựa trên các thông số như thời gian độ trễ, tiêu thụ điện năng, phạm vi truyền tin và tốc độ đọc.** **Hạn chế của nghiên cứu là việc sử dụng dữ liệu và số lượng cảm biến.** **Các tác giả [4] đang tập trung vào các ứng dụng có tính định hướng trong công việc.** **Ứng dụng y tế khẩn cấp liên quan đến ứng dụng IoT đã được sử dụng để khám phá các phương pháp truy cập dữ liệu.** **Đối với các ứng dụng IoT khác nhau, câu hỏi liệu cơ sở dữ liệu SQL hay NoSQL là tốt hơn, vẫn chưa có câu trả lời. Các nghiên cứu được thực hiện [5] [6] với dữ liệu được lưu trữ trên đám mây. Phân tích Cơ sở dữ liệu SQL và NoSQL cho ứng dụng IoT mối quan tâm của bài báo.**

1. DATABASE MANAGEMENT SYSTEM
2. *Database Overview*

Database can be defined as a structured collection of data. The system which handles the data, transactions, problems or any other aspect of the database is the Database Management System (DBMS). Relational DBMS which use structured query language (SQL) was traditional database system. The latest trend in market is the non-relational database known as NoSQL [7]. Both these databases are equal potential competitors of each other. This study is to find out the suitable database for IoT application. Before approaching to the comparative study of databases for IoT application, basic characteristics of database are explored. These characteristics of database are given by the CAP theorem which explains Consistency, Availability and Partitioning.

Cơ sở dữ liệu có thể được định nghĩa là một bộ sưu tập dữ liệu có cấu trúc. Hệ thống xử lý dữ liệu, giao dịch, sự cố hoặc bất kỳ khía cạnh nào khác của cơ sở dữ liệu là Quản lý cơ sở dữ liệu Hệ thống (DBMS). DBMS quan hệ sử dụng có cấu trúc ngôn ngữ truy vấn (SQL) là hệ thống cơ sở dữ liệu truyền thống. Các xu hướng mới nhất trên thị trường là cơ sở dữ liệu phi quan hệ được gọi là NoQuery [7]. Cả hai cơ sở dữ liệu này đều có tiềm năng như nhau Đối thủ của nhau. Nghiên cứu này là để tìm ra sự phù hợp cơ sở dữ liệu cho ứng dụng IoT. Trước khi tiếp cận nghiên cứu so sánh các cơ sở dữ liệu cho ứng dụng IoT, cơ bản đặc điểm của cơ sở dữ liệu được khám phá. Những đặc điểm này cơ sở dữ liệu được đưa ra bởi định lý CAP giải thích Tính nhất quán, sẵn có và phân vùng.

**Consistency**: means that once an update operation is finished, everyone can read that latest version of the data from the database and such system is consistent system. While, a system in which, updated data cannot be seen by all users at once is known as eventually consistent. Availability: is achieved if the system always provides continuous operation. Availability is achieved by deploying the database as a cluster of nodes, using replication or partitioning data across multiple nodes. In such case, if one node crashes, the other nodes can still continue to work. Partition Tolerance: A database system which can operate even if one of the nodes fail or is inaccessible. This is done by redirecting all queries to the failed node to some other active node of that system.

**Tính nhất quán**: có nghĩa là một khi hoạt động cập nhật là kết thúc, mọi người có thể đọc phiên bản mới nhất của dữ liệu từ cơ sở dữ liệu và hệ thống như vậy là hệ thống phù hợp. Trong khi một Tất cả người dùng tại hệ thống, dữ liệu cập nhật không thể nhìn thấy một lần được gọi là cuối cùng phù hợp. Sẵn có: là đạt được nếu hệ thống luôn cung cấp hoạt động liên tục. Tính khả dụng đạt được bằng cách triển khai cơ sở dữ liệu dưới dạng cụm của các nút, sử dụng sao chép hoặc phân vùng dữ liệu trên nhiều hạch. Trong trường hợp như vậy, nếu một nút gặp sự cố, các nút khác có thể vẫn tiếp tục làm việc Dung sai phân vùng: Một hệ thống cơ sở dữ liệu có thể hoạt động ngay cả khi một trong các nút bị lỗi hoặc là không thể truy cập Điều này được thực hiện bằng cách chuyển hướng tất cả các truy vấn đến nút không thành công cho một số nút hoạt động khác của hệ thống đó.

The traditional database systems like SQL, focused on consistency and supports following ACID properties.

Các hệ thống cơ sở dữ liệu truyền thống như SQL, tập trung vào tính nhất quán và hỗ trợ sau các thuộc tính ACID.

**Atomicity**: Partially accomplished transactions are discarded. Transaction can only be successful or unsuccessful. Consistency: In the event of transaction failure, system reverts the transaction and goes back to the previous stable state. And hence, system always remains stable. Isolation: Transactions are completed without any interference and are processed independently. Durability: All the committed transactions are saved in logs and will not be lost. This helps to recover the system in case of abnormal terminations.

**Nguyên tử**: Các giao dịch đã hoàn thành một phần là bỏ đi. Giao dịch chỉ có thể thành công hoặc không thành công. Tính nhất quán: Trong trường hợp giao dịch bị lỗi, hệ thống hoàn nguyên giao dịch và trở lại trạng thái ổn định trước đó. Và do đó, hệ thống luôn ổn định. Cô lập: Giao dịch được hoàn thành mà không có sự can thiệp nào và được xử lý độc lập. Độ bền: Tất cả các giao dịch đã cam kết là lưu trong nhật ký và sẽ không bị mất. Điều này giúp phục hồi hệ thống trong trường hợp chấm dứt bất thường.

On the other hand, NoSQL database systems focus more on Availability and Partitioning and give eventual consistency. These systems follow BASE properties.

Mặt khác, các hệ thống cơ sở dữ liệu NoQuery tập trung nhiều hơn về Tính khả dụng và Phân vùng và cung cấp tính nhất quán cuối cùng. Các hệ thống này tuân theo các thuộc tính BASE.

Basic Availability: NoSQL database focuses on availability of data as per CAP theorem requirements. Soft State: State of the database system is dynamic and may change over time due to eventual consistency. All the replicas of database do not have to be consistent all the time.

Eventual Consistency: After any Write or update or delete operation, system may not immediately reflect the modifications done. But, eventually it will become consistent showing the modified data in all replicas.

Tính khả dụng cơ bản: Cơ sở dữ liệu NoQuery tập trung vào tính sẵn có của dữ liệu theo yêu cầu định lý CAP. Mềm mại Trạng thái: Trạng thái của hệ thống cơ sở dữ liệu là động và có thể thay đổi theo thời gian do tính nhất quán cuối cùng. Tất cả các bản sao của cơ sở dữ liệu không phải lúc nào cũng nhất quán.

Cuối cùng Tính nhất quán: Sau mọi thao tác Viết hoặc cập nhật hoặc xóa, hệ thống có thể không phản ánh ngay lập tức các sửa đổi được thực hiện. Nhưng, cuối cùng nó sẽ trở nên nhất quán cho thấy dữ liệu sửa đổi trong tất cả các bản sao.

The system following BASE properties is not strictly consistent. But, all the updates or modifications will be available eventually. In such systems, clients may encounter an inconsistency in data when updation or replication process is in progress. But, the data will reach the expected consistent state after the completion of replication. In IoT application domain, Number of users, speed requirements and user demands are ever increasing and hence partition tolerance of database becomes the important characteristic to be considered. NoSQL database gives priority to availability over consistency. On the other side, SQL follows the reverse order. The aim is to find out which of these two databases will perform better for a small scale IoT application of water sprinkler system.

Hệ thống theo các thuộc tính BASE không nghiêm ngặt thích hợp. Nhưng, tất cả các bản cập nhật hoặc sửa đổi sẽ là cuối cùng có sẵn. Trong các hệ thống như vậy, khách hàng có thể gặp phải sự không nhất quán trong dữ liệu khi quá trình cập nhật hoặc sao chép đang trong quá trình Nhưng, dữ liệu sẽ đạt được sự nhất quán nhà nước sau khi hoàn thành nhân rộng. Trong ứng dụng IoT tên miền, số lượng người dùng, yêu cầu tốc độ và người dung nhu cầu ngày càng tăng và do đó dung sai phân vùng của

cơ sở dữ liệu trở thành đặc điểm quan trọng xem xét. Cơ sở dữ liệu NoQuery ưu tiên cho tính khả dụng hơn Tính nhất quán. Mặt khác, SQL tuân theo thứ tự ngược lại. Mục đích là để tìm ra cơ sở dữ liệu nào trong hai cơ sở dữ liệu này hoạt động tốt hơn cho ứng dụng IoT quy mô nhỏ của nước hệ thống phun nước.

1. *SQL Vs NoSQL from IoT perspective*

SQL Database: SQL Database follows relational data model to store the data. In this model, data is stored in rows and columns in a tabular form. Related tables can be interlinked together. Various relational databases available are MySQL, Oracle, SQLServer, etc.

Cơ sở dữ liệu SQL: Cơ sở dữ liệu SQL theo dữ liệu quan hệ mô hình để lưu trữ dữ liệu. Trong mô hình này, dữ liệu được lưu trữ trong các hang và các cột ở dạng bảng. Bảng liên quan có thể được liên kết với nhau. Cơ sở dữ liệu quan hệ khác nhau có sẵn là MySQL, Oracle, SQLServer, v.v.

NoSQL Databases: NoSQL follows non-relational data model. Non relational model supports schema-free storage of data in various forms such as document, graph. With features like horizontal scalability, schema less storage, support for unstructured data, NoSQL becomes competent for storing IoT data. Popularity for NoSQL databases spur because of the features it provides including high scalability, easy access, and distributed architecture. MongoDB, Redis, CouchDB, Hbase are some of the popular NoSQL databases available.

Cơ sở dữ liệu NoQuery: NoQuery theo dữ liệu không liên quan mô hình. Mô hình phi quan hệ hỗ trợ lưu trữ không có lược đồ dữ liệu dưới nhiều hình thức khác nhau như tài liệu, đồ thị. Với các tính năng như khả năng mở rộng theo chiều ngang, lược đồ lưu trữ ít hơn, hỗ trợ cho dữ liệu không có cấu trúc, NoQuery trở nên có thẩm quyền lưu trữ IoT dữ liệu. Sự phổ biến cho cơ sở dữ liệu NoQuery thúc đẩy vì các tính năng mà nó cung cấp bao gồm khả năng mở rộng cao, truy cập dễ dàng và kiến trúc phân tán. MongoDB, Redis, CouchDB, Hbase là một số cơ sở dữ liệu phổ biến của NoQuery có sẵn.

From IoT point of view, choosing the right database from these two database types is must. Differences between the two database types are discussed below with IoT perspective.

Từ quan điểm của IoT, chọn cơ sở dữ liệu phù hợp từ Hai loại cơ sở dữ liệu là phải. Sự khác biệt giữa hai các loại cơ sở dữ liệu được thảo luận dưới đây với quan điểm IoT.

**Scalability**: SQL database supports vertical scalability while NoSQL supports horizontal scalability. Vertical scalability refers to the ability to increase the performance of single node with adding resources such as memory or processors to the same node. In horizontal scalability, number of nodes (servers) is increased so as to share the system load. For new IoT applications which are in developing stage, using a database that has future expansion scope will be a practical choice. This will allow expansion of resources as and when required by the IoT application and no need for high initial investment.

Khả năng mở rộng: Cơ sở dữ liệu SQL hỗ trợ dọc khả năng mở rộng trong khi NoQuery hỗ trợ khả năng mở rộng theo chiều ngang. Khả năng mở rộng theo chiều dọc đề cập đến khả năng tang hiệu suất của nút đơn với việc thêm tài nguyên như bộ nhớ hoặc bộ xử lý cho cùng một nút. Theo chiều ngang khả năng mở rộng, số lượng nút (máy chủ) được tăng lên để chia sẻ tải hệ thống. Đối với các ứng dụng IoT mới trong giai đoạn phát triển, sử dụng cơ sở dữ liệu có tương lai phạm vi mở rộng sẽ là một lựa chọn thực tế. Điều này sẽ cho phép mở rộng tài nguyên theo yêu cầu của IoT ứng dụng và không cần đầu tư ban đầu cao.

**Data Retrieval**: Faster data retrieval feature will be required when user has to fetch data from database for further processing. In SQL, various tables are linked together. To lookup data from different tables, user has to use JOIN statements which creates Views. This is a time consuming process. On the other hand, in NoSQL, data is stored in form of objects which will contain all the related data. This eliminates process of combining and then displaying the data, hence saving response time.

Truy xuất dữ liệu: Tính năng truy xuất dữ liệu nhanh hơn sẽ là cần thiết khi người dùng phải tìm nạp dữ liệu từ cơ sở dữ liệu cho tiếp tục xử lý. Trong SQL, các bảng khác nhau được liên kết cùng với nhau. Để tra cứu dữ liệu từ các bảng khác nhau, người dùng phải sử dụng các câu lệnh THAM GIA tạo ra Lượt xem. Đây là một thời gian quá trình tiêu thụ. Mặt khác, trong NoQuery, dữ liệu là được lưu trữ dưới dạng các đối tượng sẽ chứa tất cả các liên quan dữ liệu. Điều này giúp loại bỏ quá trình kết hợp và sau đó hiển thị dữ liệu, do đó tiết kiệm thời gian đáp ứng.

**System Maturity**: SQL is an experienced technology and hence most of the issues have taken care of. Security features like authentication, data confidentiality & integrity are incorporated in SQL. On other hand, such security features are yet to be addressed in NoSQL. NoSQL might generate more issues or security breaches because of lack of maturity of the system. In some IoT applications, a secured communication channel is required to transfer sensitive data. For such applications, it is better to use a secured data storage system along with secured communication channel.

Trưởng thành hệ thống: SQL là một công nghệ có kinh nghiệm và do đó hầu hết các vấn đề đã được quan tâm. Bảo vệ các tính năng như xác thực, bảo mật dữ liệu và tính toàn vẹn được kết hợp trong SQL. Mặt khác, bảo mật như vậy các tính năng vẫn chưa được giải quyết trong NoQuery. NoQuery có thể tạo ra nhiều vấn đề hoặc vi phạm an ninh vì thiếu trưởng thành của hệ thống. Trong một số ứng dụng IoT, một kênh truyền thông bảo mật là cần thiết để chuyển dữ liệu nhạy cảm. Đối với các ứng dụng như vậy, tốt hơn là sử dụng một hệ thống lưu trữ dữ liệu bảo mật cùng với bảo mật kênh thông tin liên lạc.

1. SYSTEM ARCHITECTURE

For the research work, garden water sprinkler application is designed. Fig. 1 below shows the block diagram of the system. Temperature, humidity, soil moisture sensors are used in the hardware along with the water level detection. Sprinkler motor is turned ON depending on the inputs from soil moisture and water level in the tank. If the soil moisture level drops below the threshold value and if water level is TRUE, then sprinkler motor turns ON. The data from these sensors is collected and is sent to database server and user email ID. The aim is to further utilize this data for analysis using data processing tools. This IoT application provides continuous flow of data. Data is stored in both MySQL and MongoDB database systems for carrying out the comparison study.

Đối với công việc nghiên cứu, ứng dụng phun nước vườn được thiết kế. Hình 1 bên dưới cho thấy sơ đồ khối của hệ thống. Nhiệt độ, độ ẩm, cảm biến độ ẩm đất được sử dụng trong phần cứng cùng với việc phát hiện mực nước. Vòi phun nước động cơ được BẬT tùy thuộc vào các yếu tố đầu vào từ độ ẩm đất và mực nước trong bể. Nếu độ ẩm của đất giảm xuống dưới giá trị ngưỡng và nếu mực nước là TRUE, sau đó động cơ phun nước bật ON. Dữ liệu từ các cảm biến này là được thu thập và được gửi đến máy chủ cơ sở dữ liệu và ID email người dùng. Các Mục đích là để tiếp tục sử dụng dữ liệu này để phân tích sử dụng dữ liệu công cụ xử lý. Ứng dụng IoT này cung cấp liên tục lưu lượng dữ liệu. Dữ liệu được lưu trữ trong cả MySQL và MongoDB hệ thống cơ sở dữ liệu để thực hiện nghiên cứu so sánh.

Pic

1. EXPERIMENTAL RESULTS
2. *Experimental Setup*

For experimentation, hardware of water sprinkler system is designed using Arduino Uno board. Readings of soil moisture sensor and other sensors are given to analog input pins of Arduino board. This continuous reading is stored in two database systems namely: MySQL and MongoDB. MySQL version 5.5 is used for the experimentation. Along with it, MySQL graphical user interface of MySQL query browser1.2.17 is used. New schema is created to design table. Database structure created in MySQL for storing the IoT sensor data is shown below in Table I:

Để thử nghiệm, phần cứng của hệ thống phun nước là được thiết kế bằng bảng Arduino Uno. Bài đọc về độ ẩm của đất cảm biến và các cảm biến khác được trao cho các chân đầu vào tương tự của Bảng mạch Arduino. Đọc liên tục này được lưu trữ trong hai hệ thống cơ sở dữ liệu cụ thể là: MySQL và MongoDB. MySQL phiên bản 5.5 được sử dụng cho thử nghiệm. Cùng với nó, Giao diện người dùng đồ họa MySQL của truy vấn MySQL trình duyệt1.2.17 được sử dụng. Lược đồ mới được tạo để thiết kế bảng. Cấu trúc cơ sở dữ liệu được tạo trong MySQL để lưu trữ IoT dữ liệu cảm biến được hiển thị bên dưới trong Bảng I:

Table I. MySQL database structure

|  |  |
| --- | --- |
| **Fields Data-Type** | **Fields Data-Type** |
| Sensor\_id Int(40) | Sensor\_id Int(40) |
| Humidity Varchar(10) | Humidity Varchar(10) |
| Soil Varchar(10) | Soil Varchar(10) |
| Temp Varchar(10) | Temp Varchar(10) |
| Created Timestamp | Created Timestamp |
| User\_id Varchar(10) | User\_id Varchar(10) |

For mongoDB, Mongo Booster 1.6.2 client is used to create collection and documents. Connection with local MongoDB server is made. Collection which is similar to table in mysql is created. Collection name: sensor\_data is created. Documents (records) can be added in this collection using following syntax:

Đối với mongoDB, máy khách Mongo Booster 1.6.2 được sử dụng để tạo bộ sưu tập và tài liệu. Kết nối với địa phương Máy chủ MongoDB được thực hiện. Bộ sưu tập tương tự như bảng trong mysql được tạo ra. Tên bộ sưu tập: sensor\_data được tạo. Tài liệu (hồ sơ) có thể được thêm vào trong bộ sưu tập này bằng cách sử dụng cú pháp sau:

Db.collection.insert

({sensor\_id: <sensor ID number>,

Humidity: <humidity sensor reading>,

Soil\_moisture: <moisture sensor reading>,

Temperature: <temperature sensor reading>,

CreateDate: new Date(),

Userid: <User ID> })

Jmeter version 3.0 is used to measure the performances of the two database systems. Apache Jmeter is a java application. It is used for load testing functional behavior of database. Performance of the database is measured using JDBC configuration. Jmeter, MySQL, MongoDB are all open source software used in this study.

Jmeter phiên bản 3.0 được sử dụng để đo hiệu suất của hai hệ thống cơ sở dữ liệu. Apache Jmeter là một ứng dụng java. Nó được sử dụng để kiểm tra tải hành vi chức năng của cơ sở dữ liệu. Hiệu suất của cơ sở dữ liệu được đo bằng cách sử dụng JDBC cấu hình. Jmeter, MySQL, MongoDB đều là nguồn mở phần mềm được sử dụng trong nghiên cứu này.

1. *Results*

In this comparative study, response time of two databases is measured for different queries. Detailed scenarios are discussed below:

In tham số này, hợp lệ, thời gian đáp ứng hai cơ sở dữ liệu này measured for different queries. Detailed scenarios are discussed below:

1) *SELECT* query with varying number of threads:

The graph of response time Vs number of threads for SELECT query is shown below in fig. 2. Number of threads is varied from 1 to 10. Read process is carried out by using Select query which fetches 1000 records from 10000 records in the database. Response time for MySQL and MongoDB is somewhat similar till the number of threads is 4. From the graph, it is observed that, at thread 10, the rise in response time of MongoDB is increasing as compared to MySQL. In MongoDB, too many threads can overwhelm the system. This adds to the latency and eventually starving the CPU.

1. *SELECT* query with varying number of threads:  
   Các graph của thời gian câu trả lời Vs số của phiên bản cho SELECT query is shown below trong fig. 2. Number of threads is varied from 1 to 10. Read process is carried out by using Select query which fetches 1000 records from 10000 records in the database. Response time for MySQL and MongoDB is tương thích tương thích để số của phần đầu là 4. From the graph, it is observed that, at thread 10, the rise in response thời gian của MongoDB is growing as so sánh với MySQL. Print MongoDB, too many threads can overwhelm the system. This thêm vào cái latency và tương lai starving của CPU.

PIC xxx

2) *INSERT* query with varying number of threads:

The graph of response time Vs number of threads for write latency is shown below in fig. 3. For Insert Query operation, the rate of increase of response time for Mysql is more as compared to MongoDB. For insert operation, clearly MongoDB is faster.

1. *INSERT* query with varying number of threads:  
   Các graph của thời gian đáp ứng số số của threads cho ghi latency is shown below in fig. 3. For Insert Query operation, kích cỡ của tăng tăng, thời gian trả lời cho Mysql là nhiều với compared to MongoDB. For insert operation, clearly MongoDB is faster.

Pic xxx

3) *SELECT* query with varying number of records:

For SELECT query, graph of response time Vs varying number of records is plotted and is given below in fig. 4. Select query operation for varying number of records shows bit increasing trend line for MongoDB while for MySQL, it is slightly decreasing trend line. For higher number of records MySQL is also stable as compared to MongoDB as we can see drastic changes in response time of MongoDB for 9000 to 10000 records. From graph 1 and 3, it can be said that, for select query, MongoDB response time is more.

1. *SELECT* query with varying number of records:  
   For SELECT query, graph of response time Vs varying số của mục nhập là plotted và là đưa ra dưới đây trong fig. 4. Chọn câu lệnh hành động cho số thứ tự số hiệu số mục ghi bit Increasing trail trend cho MongoDB trong khi MySQL, nó là slightly decreasing trend line. For higher number of records MySQL is stable drastic changes in thời gian response của MongoDB for 9000 để 10000 records. From 1 and 3 graphs, it can be said that, for select query, MongoDB response time is more.

Pic xxx

4) *INSERT* query with varying number of records:

In the graph shown below in fig. 5, steady line of Mysql response time can be observed. On the other hand, rapid variations in MongoDB are observed. During INSERT operation in MongoDB, variations are observed as sometimes write queue is full and then insert queries are stacked up and response or latency increases. Also, when simultaneous Insert operation is going on, db is locked for that period of time. In this case also, insert can be delayed.

1. *INSERT* query with varying number of records:  
   In bản graph để hiển thị dưới đây fig. 5, steady line of Mysql thời gian đáp thể được observe. On the other hand, rapid variations in MongoDB are observed. During INSERT In MongoDB operation, variations are observed as sometimes đang đăng nhập hàng phục vụ là đầy đủ và đó thay đổi câu hỏi được stacked up và response or latency increases. Also, when simultaneous insert thao tác sẽ được tiếp tục, cơ sở dữ liệu bị khoá cho thời gian của thời gian. Print this case also, insert can be delayed.

Pic xxx

VI. CONCLUSIONS

Performance of MySQL and MongoDB databases is compared for the data generated by an Internet of Things application. MySQL which is a traditional SQL database and MongoDB which is a NoSQL database are used to store the IoT data. The comparison study is based on the time taken to execute Select and Insert queries against varying number of records and threads. As, the number of records / size of the table increases, load on the system increases, further increasing the latency or response time. Each database has its own pros and cons. From the study, it was observed that in some scenario, MongoDB required less response time compared to MySQL. But, MySQL responses were stable as compared to MongoDB. Therefore, choosing a better database for IoT depends on which query is mostly used and the requirements of application. Future research run for this work include investigation performed with several other database management systems available with advanced load testing tools. Research can be further extended by using cloud, distributed databases which will make the work more realistic.

VI. CONCLUSIONS  
Hiệu lực của MySQL and MongoDB databases is compared để tạo dữ liệu bởi một ứng dụng Internet  
MySQL này là một cơ sở dữ liệu cơ sở dữ liệu và MongoDB mà là một cơ sở dữ liệu khôngSQL dùng để lưu dữ liệu IoT. The chế độ sự so sánh được trên thời gian đã thực hiện để thực hiện Select  
và Hãy nhập câu lệnh với số thứ tự số thứ tự và mục ghi threads. As, số của mục ghi / kích cỡ của sự nâng cấp bảng, tải về sự tăng hệ thống, tiếp tục tạo hoặc response time. Each cơ sở dữ liệu có trình chính không và cons. From sự học, nó đã xác định trong một số mẫu, MongoDB cần thiết ít đối số thời gian dùng để MySQL. Nhưng, các phản hồi của MySQL vẫn ổn định so với MongoDB. Do đó, việc chọn cơ sở dữ liệu tốt hơn cho IoT phụ thuộc vào truy vấn nào được sử dụng nhiều nhất và các yêu cầu của ứng dụng. Nghiên cứu trong tương lai cho công việc này bao gồm điều tra được thực hiện với một số cơ sở dữ liệu khác hệ thống quản lý có sẵn với thử nghiệm tải tiên tiến công cụ. Nghiên cứu có thể được mở rộng hơn nữa bằng cách sử dụng đám mây, cơ sở dữ liệu phân tán sẽ làm cho công việc thực tế hơn.

Tài liệu tham khảo

[1] s. m. riazul islam, daehan kwak, md. humaun kabir, Mahmud hossain and kyung-sup kwak1, “The Internet of Things for Health Care: A Comprehensive Survey,” IEEE Access, June 2015.

[2] Satyadhyan Chickerur, Anoop Goudar, Ankita Kinnerkar, “Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications,” 2015 8th International Conference on Advanced Software Engineering & Its Applications(ASEA), Jeju, pp.41-47, 2015.

[3] Y. Su, A. Wickramasinghe and D. C. Ranasinghe, "Investigating sensor data retrieval schemes for multi-sensor passive RFID tags," 2015 IEEE International Conference on RFID (RFID), SanDiego, CA, pp.158-165, 2015.

[4] Xu, L. D. Xu, H. Cai, C. Xie, J. Hu and F. Bu, "Ubiquitous Data Accessing Method in IoT-Based Information System for Emergency Medical Services,” in IEEE Transactions on Industrial Informatics, vol.10, no.2, pp.1578-1586, May 2014.

[5] Hammes, Dayne; Medero, Hiram; and Mitchell, Harrison, "Comparison of NoSQL and SQL Databases in the Cloud" SAIS 2014 Proceedings, 2014.

[6] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on,

Victoria,BC, pp.15-19, 2013.

[7] Ion LUNGU, Manole VELICANU, Iuliana BOTHA, “Database Systems – Present and Future,” Informatica Economică vol. 13, no.1/2009