Journal of Engineering and Applied Sciences 13 (Special Issue 3): 3278-3283, 2018
ISSN: 1816-949X

NoSQL vs. SQL: Operations Cost Evaluation; A Case Study on a Social Network Application

[1, 2]Minh-Son Dao, [2]Yen-Nhi Tran-Thi, [2]Van-Cuong Mai and [2]Tuan-Anh Nguyen-Gia
[1]Universiti Teknologi Brunei, Jalan Tungku Link, BE1410 Gadong, Brunei
[2]University of Information Technology, Ho Chi Minh City, Vietnam

Abstract: The era of big data coming from everyone, everything and everywhere raises a big challenge to data science, especially, the database category. Meanwhile, SQL databases guarantee ACID (Atomicity, Consistency, Isolation, Durability) and require an expensive budget when scaling-out a system, NoSQL databases can vary from BASE (Basically Available, Soft state, Eventual consistency) to ACID and require less budget than SQL databases for scalability (both in hardware and software). The argument of choosing which databases for applications that run on big data, especially, social network applications has still maintained open questions. This study aims to contribute a practical discussion such questions by introducing a brief-but-concrete comparison between SQL and NoSQL databases and among four categories of NoSQL databases (Key-value, document, column, graph) under practical perspective. Moreover, the study also evaluates the operations cost on four basic operators (Insert, select, update, delete) running on a small-scale social network application with SQL and NoSQL databases. The experimental results confirm that NoSQL can be a good candidate for big data applications, especially, social network applications where select and insert operators are the utmost requirements.

Key words: SQL, NoSQL, database, operations cost, social networks, applications

INTRODUCTION

For more than three decades, relational databases play an important role to manage transaction-oriented applications (Darwen, 2014). Relational databases are also well-known as a database type with a very high transaction reliability because they fully support the four properties of transactions: Atomicity, Consistency, Isolation and Durability (ACID) (Mohamed *et al*., 2014). Unfortunately, one of so-called weak points of relational databases when dealing with large-scale data is the scalability. Cattell (2011) claims that, although, relational databases scale well on a single server (i.e., "scale-up"), they will start to cause problems when the capacity of a single server is reached and the data needs to be distributed on multiple servers (i.e., "scale-out"). Eventually, the complexity of scaling to a huge amount of servers becomes overwhelming and the money needs to be spent in order to support expensive hardwares, softwares and labors becomes unbearable to the users.

We are now living in the era of data that comes from everywhere, everyone and everything. The data are not simply in the text format but in heterogeneous formats which varied from text, audio, image and video, to name just a few. Therefore, the emerging requirement of storing and manipulating efficiently and effectually such heterogeneous data (i.e., big data) becomes the utmost consideration in these days (Zala and Dhobi, 2015).

Several discussions have been done to understand whether the relational databases can be able to cope with the rapid growth of data in both quantity and quality aspects as mentioned above. These discussions come up with the common answer that there should be other types of databases that can cope with data's tsunami (Han *et al*., 2011).

NoSQL databases are now known as one of the answers. There are several popular NoSQL databases that have been used by both commercial and academic purposes such as MongoDB, HBase and Neo4J.

These databases are classified into four categories: key-value, graph, column and document stores (Loureno *et al*., 2015). Meanwhile, relational databases guarantee that the ACID, NoSQL databases can vary from BASE (Basically Available, Soft state and Eventual consistency) to ACID. That gives NoSQL the flexible ability to cope with several significant problems of big data such as transaction reliability, data model, scalability, cloud, big data handling, data warehouse, complexity, crash recovery and security, to name a few

Correponding Author: Minh-Son Dao, Universiti Teknologi Brunei, Jalan Tungku Link, BE1410 Gadong, Brunei

(Gudivada *et al*., 2014). Moreover, NoSQL databases can support the development of a new generation of webs, social networks and cloud-computing that require a high scalability, a high availability, efficient big data storage and access requirements, high concurrency of reading and writing with low latency and flexible and economic expansion ability (Sullivan, 2015; Pokorny, 2013; Kanoje *et al*., 2015; Kim, 2014).

In order to contribute to the answer of whether using SQL and/or NoSQL for big data, especially, social network applications that have developed rapidly these days, this study aims to introduce a brief but concrete comparison between SQL and NoSQL databases under practical perspective. Moreover, the evaluation of operations cost of SQL (e.g., Microsoft SQL Server) and NoSQL (e.g., MongoDB) databases is investigated as well. A case study is built as a social network application where users can post, modify and share their data both in image and text data. Besides, they can make-friends/un-friends and exchange their data among a friend network. In fact, the case study mimics Facebook's application but on a smaller scale. The four important operators of database manipulation (Insert, select, update, delete) are thoroughly investigated on a real time mode with a huge data generated frequently. The experimental results confirm that NoSQL databases suits practically for applications that require big data and focus on insert and select operators.

Literature review: Several surveys of SQL and NoSQL as well as comparisons between SQL and NoSQL have been done along with the development of NoSQL and big data (Darwen, 2014; Mohamed *et al*., 2014; Zala and Dhobi, 2015; Han *et al*., 2011).

Darwen *et al*. (2004) made a comprehensive survey for giving a deep insight of SQL databases including their history, data type, scalability, availability and consistency, to name but a few. Mohamed *et al*. (2014) introduce a comparison between SQL and NoSQL databases focusing on their availability, scalability and distributed ability for web applications. Security is also discussed as the main topic in this comparison. Zala and Dhobi (2015) also discuss scalability and security requirements of data mining when using NoSQL databases and how Hadoop with its MapReduce scheme can reduce the time for mining data managed by NoSQL databases. Lourenco *et al*. (2015), differ from other surveys or comparisons based on benchmarks (data and systems), focus on scenarios that really happen in the real world. In their research, a concise and up-to-date comparison of NoSQL engines are gathered. Then, from the software engineer viewpoint, their advantages and drawbacks are discussed thoroughly. Gudivada *et al*. (2014) argue the need for out-of-the-box horizontal scalability for data management systems, especially, big data management under the umbrella term NoSQL. Several discussions are made in this study to choose an appropriate NoSQL system for a given bigdata application.

Following the approach introduced by Lourenco *et al*. (2015) this study compares SQL and NoSQL databases under practical perspective that aims to give the readers a quick but comprehensive insights for them to select a suitable type of database for their applications. Moreover, we build a real scenario based on social network application schema to evaluate the operations cost of both SQL and NoSQL databases.

MATERIALS AND METHODS

NoSQL vs. SQL; A brief comparision: In this study, a brief comparison between SQL and NoSQL on certain criteria is introduced. Then four categories of NoSQL databases are discussed comprehensively.

Comparing SQL and NoSQL from the practical perspective: Table 1 denotes a brief comparison between SQL and NoSQL databases on database type, developing purpose, popular dbms, data modeling, scaling, source code, support, data manipulation and consistency criteria. These criteria are always paid attention to when looking for a suitable type of database to build applications. From this brief comparison, users can go deeply inside each type of databases for further questions as follows.

Cost and scalability: SQL databases have a big problem with scalability, especially, "scale-out" (i.e., distributed system with multiple servers). That puts the burden of buying expensive hardware (both servers and storage systems) and supportive software as well as a re-training fee for labors. On the opposite side, NoSQL databases use clusters of cheap commodity servers and open source codes that can alleviate the burden of budgets. One of significant characteristics of NoSQL databases is auto-sharding (i.e., the ability of spreading automatically data across servers without stopping running applications). This can make NoSQL databases better for horizontally scaling.

Speed and heterogeneous data: The join operator that plays an important role in SQL databases under a saving-storage-space perspective can be ignored in NoSQL. By using different data structures (e.g., JSON) and without joining tables, many operations will run faster

Table 1: NoSQL and SQL databases comparison (on certain criteria)

Types	SQL	NoSQL
Database type	Different forms	Key-value, document, column and graph databases
Developing purpose	For applications that require storing data instantly and initially	For applications that require large-scale and heterogeneous data, especially, data without fixed structures
Popular DBMS	Microsoft SQL server, MySQL, PostGre, Oracle and IBM DB2	MongoDB, DocumentDB, Cassandra, Coachbase, HBase, Redis and Neo4J
Data modelling	Storing structures before data	Storing data without building structures beforehand
Scaling	Vertically scaling	Vertically and/or horizontally scaling with cloud-computing supports
Source code	Open and close source codes	Only open source codes
Support	Good	Limited supports
Data manipulation	SQL-based querying	Object-oriented APIs
Consistency	High	Strong consistency (e.g., MongoDB), eventually consistency (e.g., Cassandra)

Table 2: Database types, related DBMS and key criteria for using NoSQL

NoSQL databases	Key criteria for using NoSQL (being briefed from 8)
Key-value	
Voldemort	Caching data from relational databases to improve performances
Redis	Tracking transient attributes in Web applications such as shopping carts
Membase	Storing configuration and user data information for mobile applications
	Storing large objects such as images and audio files
Document	
Riak	Back-end support for websites with high volumes of read and write operations
MongoDB	Managing data types with variable attributes such as products
CouchDB	Tracking variable types of metadata
	Applications that use JSON data structures
	Applications benefitted from de-normalization by embedding structures within structures
Column	
Cassandra	Applications that require the ability to always write to the database
HBase	Applications that are geographically distributed over multiple data centers
Hypertable	Applications that can tolerate some short-term inconsistency in replicas
	Applications with dynamic fields or with the potential for truly large volumes of data such as hundreds of terabytes
Graph	
Sesame	Networks and IT infrastructure management
BigData	Identity and access management
Neo4J	Business process management
GraphDB	Products and services recommendation
FlockDB	Social networks

in NoSQL databases than SQL database. Specifically, the JSON format, a schemaless format, allows a database to easily and quickly import/export any new type of data without being disrupted by content structure changes.

Availability: With NoSQL databases, servers can be manipulated (e.g., add or remove) without any serious influence on a time processing. Besides, most NoSQL databases support data replication. Data can be stored with multiple copies across the cluster or even data centers. This can ensure the high availability and disaster recovery of systems which use NoSQL databases.

Database types, related DBMS and key criteria for using NoSQL: Table 2 illustrates a brief introduction of NoSQL databases categories, related DBMS and key criteria for selecting NoSQL databases (Sullivan, 2015).

Table 3 denotes a comparison among some DBMS of four categories of NoSQL databases under querying, concurrency control, partition tolerance and replication criteria.

Operations cost evaluation: In this study, a so-called social network small application is built to evaluate operations cost between SQL and NoSQL databases. The application can be considered as a small-scale-copy of Facebook where users can create their account, post and update their information (text and image data) on their own page, add comments, likes and shares information on their own and their friend's pages and can make-friend or un-friend with other users. Lecturers and students are joined as volunteers to generate data.

Objects descriptions: These objects depicted in Table 4 are described in details as follows; User = {id: ObjectId, e-Mail: string, password: string}: each user has its own (id, email, password).

UserInfo = {id: ObjectId; UserName: string; Picture: string; Mobilephone: string; Birthday: date; Gender: Int}: each user has their own user's information (i.e., Profile). Album = {id: ObjectId, Name: string; Status: string; UserId: ObjectId; CreateDate: date}: each album is denoted by (id, album's name, album's sharing status, album's owner, album's created date).

Table 3: NoSQL databases comparisons

	Querying		Concurrency control		Partition tolerance		Replication	
NoSQL DBs	Java API	Querying language	Locking	Optimistic locking	Ranging	Hashing	Read	Write
Key-value								
Redis	Yes	No	No	Yes	No	Yes	No	No
Membase	Yes	No	No	Yes	No	Yes	No	No
Document								
MongoDB	Yes	No	No	No	Yes	No	No	No
CouchDB	Yes	No	No	No	No	Yes	Yes	Yes
Column								
Cassandra	Yes	Yes	No	No	No	Yes	No	No
Hbase	Yes	Yes	Yes	No	Yes	No	No	No
Graph								
Neo4J	Yes	Yes	Yes	No	No	No	Yes	Yes
GraphDB	Yes	No	No	No	No	No	No	No

Table 4: Objects descriptions

Objects	Descriptions
User	User's login activities
UserInfo	User's profile
Post	User's posts
Comment	User's post-related comments
Album	User's album information
UserRequest	User's make-friend/un-friend requests
Friends	User's friends
Like	"Like" action logs
TimeLine	Post on "Timeline" logs

Photo = {id: ObjectId; Name: string; ReferenceId: ObjectId; Href: string; CreateDate: date; LastModify: date; Type: Int}: each photo is determined by (photo's id, photo's name, photo's comments, photo's link, photo's created date, photo's last modified date, photo's publishing type).

FriendRequest = {id: ObjectId, UserRequestId: ObjectId, UserResponseId: ObjectId}: Each FriendRequest is determined by (its ID, ID of person who requests to be a friend, ID of person who is requested to grant a friend relation).

Friend = {id: ObjectId; UserId1: ObjectId; UserId2: ObjectId; Type: Int}: friend is created to store list of friends. Each record of the friend object contains user's ID (UserId1) and the ID of his/her friend (UserId2). Besides, type is used to describe the level of friend's relation (e.g., best friend, mutual friend, friend in common, etc.).

Like = {id: ObjectId, ReferenceId: ObjectId, UserId: ObjectId; CreateDate: date; LastModify: date, Type: Int}: Each Like is invoked by a user needs to be managed by its own ID; ReferenceId refers to the content which is liked; UserId let us know who invoked "Like" and CreateDate and LastModify are used to control the date.

Post = {id: ObjectId, name: string; Content: string; UserId: ObjectId; CreateDate: date; LastModify: date}: each post is distinguished from others by its own ID, title (i.e., Name), content (i.e., Content), author/writer (i.e., UserId), created date (i.e., CreateDate) and last modification date (i.e., LastModify).

Comment = {id: ObjectId; Content: string; ReferenceId: ObjectId; CreateDate: date; LastModify: date, UserId: ObjectId}: each comment made by a user needs its ID, content (i.e., Content), created date (i.e., CreateDate), the last modified date (i.e., LastModify) and who wrote it (i.e., UserId).

TimeLine = {id: ObjectId, ReferenceId: ObjectId, UserId: ObjectId; CreateDate: date; LastModify: date; ObjectType: Int; SecurityType: Int; WallId: ObjectId}: each timeline is used to manage user's information that are posted by users themselves on their own or other user's main page. In order to do that, we need TimeLine ID, TimeLine's owner (i.e., UserId), TimeLine's created date and the last modification (i.e., CreateDate and LastModify). ObjectType and SecurityType are used to control the type and the level of security of TimeLine, respectively. Finally, WallId lets us know whose main page the content of TimeLine is posted.

RESULTS AND DISCUSSION

All data generated by the application and volunteers are stored and used for evaluation on a specific system as described in Table 5.

We then create two different databases: SQL database with the Entity-Relation Diagram (ERD) as illustrated in Fig. 1. It is clear to see that so many JOINs are created to satisfy the 3rd normal form. NoSQL database where each object is determined in JSON format. There is no JOIN created among JSON files.

Figure 2-5 illustrate the results of this evaluation on insert, select, delete and update operators, respectively. It should be noted that these diagrams only show the average values that are accumulated by three different runs (on the same databases).

The experimental results confirm that the operations cost of insert and select operators of NoSQL databases is significantly better than SQL databases. With Delete operators, it seems there is a certain threshold (in our

Table 5: Hardware and Software configuration

Objects	Parameters
Operation system	Window 8.1 pro 64 bits
CPU	Intel©CoreTMi5–2430 M CPU 2.4 GHz
RAM	8 GB
HDD	500 GB, 7200 rpm, SATA
Database	MongoDB 3.0, SQL Server 2014 Express, GraphDB
GUI	Robotmongo 0.8.5, SQL Server 2014 Management Studio, GraphDB

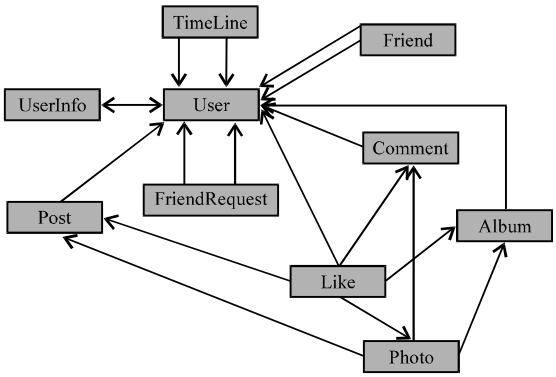

Fig. 1: ERD for SQL database

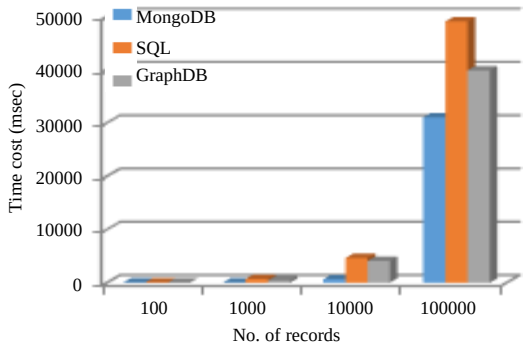

Fig. 2: Insert operation cost evaluation

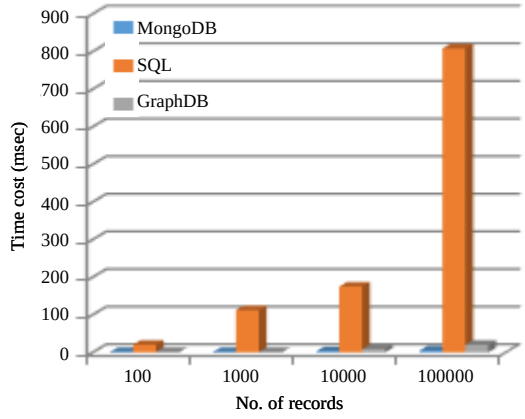

Fig. 3: Select operation cost evaluation

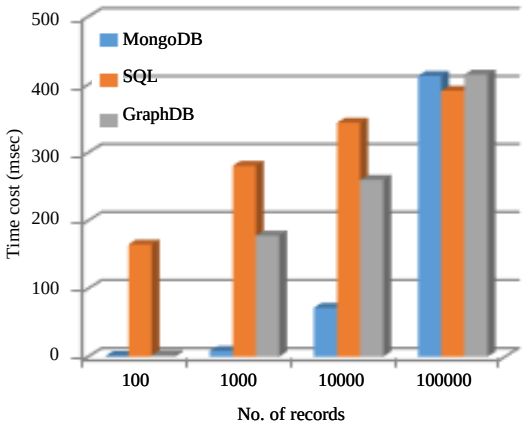

Fig. 4: Delete operation cost evaluation

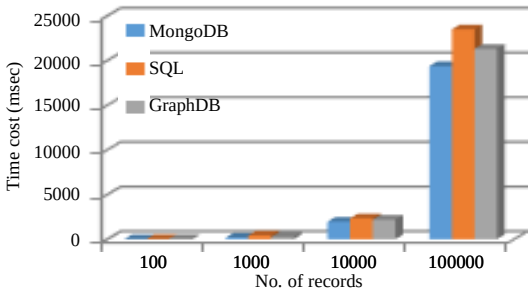

Fig. 5: Update operation cost evaluation

experience, the threshold is set as 100,000). If the number of deleted records exceeds this threshold, SQL databases can work better than NoSQL databases. Nevertheless, with a cloud-storage schema, we can definitely divide our NoSQL databases into several small storages of contained data less than the threshold, so that, the delete operator can work faster. For example as illustrated in Fig. 4, if we divide our NoSQL database to several stores, each contains a maximum of 10,000 objects, then the delete operator works triple time faster than SQL database.

CONCLUSION

This study introduces a brief-but-concrete comparison between SQL and NoSQL databases and among four categories of NoSQL databases under practical perspective. Moreover, the study evaluates operations cost for four basic operators (Insert, select, update and delete) applied on a small-scale social network application. The experimental results confirm that NoSQL databases (in this study, MongoDB and GraphDB are used as a representative) work superior than SQL

databases (in this study, Microsoft SQL server is used as a representative), especially, social network applications.

RECOMMENDATION

In future, the researchers will investigate on more operators as well as check scalability and availability when data reach storage limitation and require horizontally scaling.

REFERENCES

Cattell, R., 2011. Scalable SQL and NoSQL data stores. ACM. Sigmod Rec., 39: 12-27.

Darwen, H., 2014. SQL: A Comparative Survey. 2nd Edn., Bookboon, London, UK., ISBN:978-87-403-0778-8, Pages: 169.

Gudivada, V.N., D. Rao and V.V. Raghavan, 2014. NoSQL systems for big data management. Proceedings of the 2014 IEEE World Congress on Services, June 27-July 2, 2014, IEEE, Huntington, West Virginia, ISBN:978-1-4799-5070-6, pp: 190-197.

Han, J., E. Haihong, G. Le and J. Du, 2011. Survey on NoSQL database. Proceedings of the 6th International Conference on Pervasive Computing and Applications, October 26-28, 2011, Port Elizabeth, pp: 363-366.

Kanoje, S., V. Powar and D. Mukhopadhyay, 2015. Using MongoDB for social networking website deciphering the pros and cons. Proceedings of the 2015 International Conference on Innovations in Information Embedded and Communication Systems (ICIIECS) 2015, March 19-20, 2015, IEEE, Pune, India, ISBN:978-1-4799-6817-6, pp: 1-3.

Kim, W., 2014. Web data stores (aka NoSQL databases): A data model and data management perspective. Intl. J. Web Grid Serv., 10: 100-110.

Lourenco, J.R., V. Abramova, M. Vieira, B. Cabral and J. Bernardino, 2015. NoSQL Databases: A Software Engineering Perspective. In: New Contributions in Information Systems and Technologies, Rocha, A., A. Correia, S. Costanzo and L. Reis (Eds.). Springer, Berlin, Germany, pp: 741-750.

Mohamed, M.A., O.G. Altrafi and M.O. Ismail, 2014. Relational vs. nosql databases: A survey. Intl. J. Comput. Inf. Technol., 3: 598-601.

Pokorny, J., 2013. NoSQL databases: A step to database scalability in web environment. Intl. J. Web Inf. Syst., 9: 69-82.

Sullivan, D., 2015. NoSQL for Mere Mortals. Addison-Wesley, Boston, Massachusetts, ISBN-13: 978-0-13-402321-2, Pages: 513.

Zala, C.M. and S.J. Dhobi, 2015. A survey on data mining and analysis in Hadoop and MongoDB. Comput. Eng. Intell. Syst., 6: 11-18.