SQL vs NoSQL: A study based on Oracle and MongoDB

Chenglong Ma
s3629862

I. Introduction

This report briefly compared the performance of Oracle and MongoDB database, two representative databases of SQL and NoSQL respectively from the aspects of *query performance*, *resource requirements* and *security issues*, etc. To provide a complete and fair comparison, we first setup our experiment environment. Then, we discuss all aspects mentioned in separate sections.

II. Query Performance

We set a series of experiments for query performance, i.e. "CRUD" (**C**reate **R**ead **U**pdate **D**elete) [1] operations, between Oracle and MongoDB.

Experiment Setup

Based on the implementations of *Facebook-Lite* using Oracle and MongoDB databases, we set three "`Member`" tables/ collections (called "datasets" below for convenience) named `Member_S`, `Member_M` and `Member_L`, and they consist of $10^3, 10^4, 10^5$ records respectively. Should note that, we conducted the "**C**reating" experiments during constructing the experiment environment. Then we execute different groups of database query statements to compare their performance. All experiments are conducted multiple times to get stable test results.

A. Creating

During environment construction, the executing time was respectively recorded when $10, 10^2, 10^3, 10^4, 10^5$ members created. The result shown in Table I and Fig. 1 indicates that Oracle database is more efficient to dealing with little amount of data while MongoDB, on the contrary, outperforms Oracle on large amount.

TABLE I: Creating Time (msec)

No. of members	Oracle DB	MongoDB
10	37	786
10^2	41	5
10^3	1546	40
10^4	8864	705
10^5	83374	4368

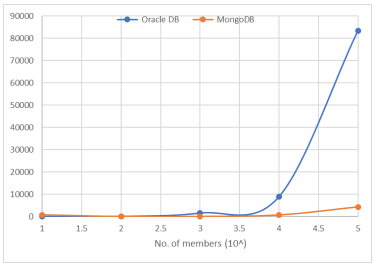

Fig. 1: The line chart of Creating Time (msec)

TABLE II: Reading Time (msec)

Datasets	Read All		Read by Index		Read by Non-index	
	Oracle DB	MongoDB	Oracle DB	MongoDB	Oracle DB	MongoDB
Member_S	304	0.1	156	0.1	205	0.1
Member_M	402	3	202	0.1	254	5
Member_L	455	35	256	0.1	258	37

B. Reading

In this section, we conduct three sub-jobs on all datasets, i.e. reading all data, reading by index field, reading by non-index field. The result are shown in Table II.

It is obvious that MongoDB excels Oracle in all operations by orders of magnitude. Longitudinally, reading data by index is much faster than by non-index.

C. Summary

We omit the two sections of "**Updating**" and "**Deleting**" as this report is subject to space and MongoDB has shown great advantages over Oracle for all query operations. Ansari believed that the performance of SQL-based databases is stabler than NoSQL [2] which can be a reference but not evaluated in this report.

III. Resource Requirements

Oracle databases allocate space quotas based on a logical structure that splits data into *tablespaces*, *segments*, *extents* and *blacks* [3]. They help Oracle to backups and recoveries more efficiently and manage data space effectively. MongoDB, the typical NoSQL database, is equipped with non-relational structure providing more flexible data management scheme, especially the horizontal scalability [4].

The following table III shows the actual disk usages of each dataset in Oracle and MongoDB.

It is visible that Oracle and MongoDB cost approximately the same disk space. However, Bryla and Loney demonstrated

TABLE III: Disk usage (MB)

Datasets	Oracle DB	MongoDB
Member_S	0.07	0.06
Member_M	0.75	0.65
Member_L	8.11	6.7

in [3] that Oracle has provided compression techniques to reduce the disk cost. The compressed data size may reach 50% of table III, but it requires more RAM and CPU to access full size data [5]. Therefore, it is considerable to achieve a trade-off balance between resource cost and performance.

IV. Security Issues

SQL injection is one of the most serious problems among relational databases. Attackers obtain authorized resources (e.g. the access permission) by injecting SQL code into input boxes of web applications [6]. The types of SQL injection mainly include Tautologies, Union Query, Stored Procedure, etc [6]. Extensive researches have devoted their efforts for countermeasures against SQL Injection, e.g. SAFELI [7] to detecting injection vulnerabilities, SQL-DOM framework [8] to identify the obstacles by building secure communications.

Pursuing better scalability and flexibility, MongoDB lacks sufficient consideration of security and consistency [9]. Even without the inherent problems, MongoDB suffers from its own security threats such as unencrypted data in storage, unencrypted communications between databases and clients, authentication/authorization issues on clusters, JavaScript-based injection attacks.

V. ACID Properties and Data Integrity

ACID (stands for Atomicity, Consistency, Isolation, Durability) properties allow databases to provide reliable transactions and keep data integrity. It has been well-developed for SQL-based databases while NoSQL databases tend to been partial supported. MongoDB started to support multi-document transactions since version 4.0 [10].

Atomicity

Oracle database implements atomicity by providing *undo* operation while MongoDB has been capable of guaranteeing data integrity by its *single-document* atomicity.

Consistency

A typical example of consistency is accounts transfer that database should keep the balance consistent before and after transfer. SQL-based databases basically full-support these operations and MongoDB implements it by so-called *all-or-nothing* execution.

Isolation

Various transactions are isolated between each other to keep data consistency. Oracle provide transaction-level isolation while MongoDB considers snapshot isolation. Errors would be issued and opeations roll back if access conflict occurs.

Durability

Durability ensures data loss would not happen. Oracle employs *redo* logs to retrieve data after system failure while MongoDB focusing on *replication and snapshotting* instead of durability [11].

Data Integrity

Besides ACID, oracle uses constraints maintenance to guarantee integrity which is MongoDB attempt to explode. Therefore, users should trade the flexibility and strong integrity when choosing databases.

VI. Scalability

Coping with increasing requests, databases provide various approaches to improve scalability and throughputs. Oracle supports vertical and horizontal scaling [3]. Vertical scalability allows Oracle to start some virtual machines based on the same settings with the single node. Horizontally, Oracle divides workload into groups of server instances [3]. Also, as mentioned in Section III, *extents* and *blacks* help Oracle to assign disk dynamically in a single server.

Compared with Oracle, MongoDB has the edge of horizontal scaling. *Sharding* technique of MongoDB is built on collection-level and allows distributing data over multiple servers [12]. Its cluster is constructed by *shard*, *mongos* and *config servers* [12]. The read/write requests are handled across different shards which reduces the pressure of the database system. Besides, this scheme increases the capacity of clusters by adding more shards. Distributing data also means sharing the risk of system failure. Thus, the Sharding mechanism has improved the availability of MongoDB [12].

However, for Oracle, it is difficult to restructure the table schema which is the edge of MongoDB. Therefore, Oracle requires the consistency of table structure when scaling. It might be indifferent for stable business but over the left for dynamical operations based enterprises, e.g. internet companies.

VII. "Big" Data

The well-developed scalability of Oracle and MongoDB has improved their ability to handle massive volumes of data as well. To deal with the big data, Oracle has provided multiple applications and strategies advancing with the times which include integration with Apache Hadoop, Oracle Big Data Appliances, Oracle Big Data Connectors [13]. The advantages of Oracle are their mature industrial big data solutions and stable cloud services. Standardization SQL-based databases can managed massive volumes of data effectively. Also, Oracle achieves outstanding performance for data warehouses which provides accurate analysis based on the big data.

By contrast, MongoDB has been equipped with the outstanding effectiveness to access the desired documents by index [14]. It can be evidenced by the experiments in Section II. MongoDB is capable of scaling database capacity and document structure. It has reduced the development difficulty of inconstant applications.

VIII. Complex Queries Execution

Oracle implements various SQL functions to extend the views of tables (e.g. `join`, `materialized view` and `PL/SQL` functions). These techniques help developers to execute complex queries easily in database-level and application-level. For example, in our *Facebook-Lite* project, `join`ing `Member`, `Post` and `Friends` tables can get a comprehensive of post-comments. However, although providing some similar techniques called "`Aggregations`", MongoDB lacks API-level support (e.g. PHP library).

We conduct experiments over three `Member` datasets to fetching their posts to simulate complex queries. The results are shown in Table IV.

TABLE IV: Complex Queries Time (msec)

Datasets	Oracle DB	MongoDB
Member_S	55	859
Member_M	86	1688
Member_L	108	2251

It can be evidenced that Oracle outperforms MongoDB for complex querying.

IX. Conclusion and Recommendations

Based on our observation and comparison, MongoDB, as a novel and flexible database system, performs outstanding edges of big data accessing and horizontal scalability. It is suitable for dynamic schemas and inconstant business, especially internet companies. Instead, Oracle, the representative of relational databases, outperforms NoSQL databases on security and ACID transaction management. It is more considerable in stable business domains which have constant data structures and require accurate and complicate analysis. Besides, it tends to be the first choice of security concerned enterprises due to its mature secure approaches.

(word count: 1523)

References

[1] J. Martin, *Managing the Data-Base Environment*. Englewood Cliffs, N.J. : Prentice-Hall, 1983, 00270 Includes bibliographical references and index.

[2] H. Ansari, "Performance comparison of two database management systems MySQL vs MongoDB," 2018, 00000.

[3] B. Bryla and K. Loney, *Oracle Database 12C the Complete Reference*. McGraw-Hill Osborne Media, 2013, 00008.

[4] S. H. Aboutorabi, M. Rezapour, M. Moradi, and N. Ghadiri, "Performance evaluation of SQL and MongoDB databases for big e-commerce data," in *2015 International Symposium on Computer Science and Software Engineering (CSSE)*, Aug. 2015, pp. 1–7, 00020.

[5] C. Farrugia, "MySQL vs. MongoDB Disk Space Usage," https://www.revulytics.com/blog/mysql-vs-mongodb-disk-space-usage, 00000.

[6] D. A. Kindy and A.-S. K. Pathan, "A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques," in *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, Jun. 2011, pp. 468–471, 00079.

[7] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao, "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1, Jul. 2007, pp. 87–96, 00141.

[8] R. McClure and I. Kruger, "SQL DOM: Compile time checking of dynamic SQL statements," in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, May 2005, pp. 88–96, 00234.

[9] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, "Security Issues in NoSQL Databases," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, Nov. 2011, pp. 541–547, 00182.

[10] M. Keep and A. Cabral, "MongoDB Multi-Document ACID Transactions are GA — MongoDB Blog," https://www.mongodb.com/blog/post/mongodb-multi-document-acid-transactions-general-availability, Jun. 2018, 00000.

[11] MongoDB, "What about Durability? — MongoDB Blog," https://www.mongodb.com/blog/post/what-about-durability, Feb. 2010, 00000.

[12] "Sharding — MongoDB Manual," https://docs.mongodb.com/manual/sharding, 00000.

[13] T. Plunkett, B. Macdonald, B. Nelson, M. Hornick, H. Sun, K. Mohiuddin, D. Harding, G. Mishra, R. Stackowiak, K. Laker, and D. Segleau, *Oracle Big Data Handbook*, 1st ed. McGraw-Hill Osborne Media, 2013, 00017.

[14] R. Lutz, P. Ameri, T. Latzko, and J. Meyer, "Management of Meteorological Mass Data with MongoDB," p. 8, 2014, 00008.