Evaluating the performance of SQL and NoSQL databases in an IoT environment

Seenauth Reetishwaree
Department of Software and Information Systems
University of Mauritius
Reduit, Mauritius
reetishwaree.seenauth@umail.uom.ac.mu

Visham Hurbungs
Department of Software and Information Systems
University of Mauritius
Reduit, Mauritius
v.hurbungs@uom.ac.mu

***Abstract*—Databases play an important role in Information and Communication Technologies and data has no significance unless it is converted into meaningful information. Relational or SQL disk databases where data is structured in tables have served many applications over the years. The increasing need for faster data access in real-time systems has changed the traditional Disk-based database into an In-memory database where data is manipulated and stored directly in the main memory thereby reducing disk input/output operations. With the emergence of IoT and Big Data, NoSQL databases now represent an alternative to traditional relational databases in terms of simpler design and faster operations on large data volumes. This work compares three types of databases namely Relational On-disk, Relational In-memory and NoSQL On-disk/In-memory in terms of Insert, Select, Update and Delete operations together with Indexes, Stored procedures and Triggers in an IoT environment. A plant environment has been monitored in real-time using sensor data such as Temperature/Humidy, Moisture and Water level. Data was stored in different databases and the time taken to execute various operations were recorded. Experiments using the water level parameter demonstrate there is no specific database that provides the best performance for all data operations.**

***Index Terms*—Database, Relational, SQL, NoSQL, IoT**

I. Introduction

In this digital era, Information and Communication Technologies have become a key component to enable firms to leverage competitive advantages in the market. Millions of business transactions are collected on a daily basis and these transactions produce terabytes of structured and unstructured data. Real-time data has enabled top management to re-align their decision towards their goals and objectives. However, the biggest challenge is that a faster analysis and processing of data is crucial for companies to remain competitive. Database management systems are very important to organisations because they control, manage and organise data.

With the emergence of new computing environments and network infrastructures such as Cloud computing, Internet of Things (IoT), Big Data and 5G, many companies and organisations are now finding themselves with the difficulty of choosing which type of database will best suit their business requirements. In IoT systems, it is important to understand the data flow: How is the data created? How is the data stored? and How is the data analysed? In a typical IoT environment, smart devices and sensors create data, which is sent via a network to a local or Cloud storage where all the data is centralised and organised. The data is then analysed to look for trends over time. Time-series data provide more value such that routine tasks can be automated based on a certain set of conditions. For example, regulate traffic lights based on congestion level, observe plant growth by analysing environment parameters and patient monitoring using wearable sensors among others. Therefore, important data considerations are: access control, performance, sharing, security, data redundancy and data integrity.

This paper is organised as follows: Section II provides a brief overview of SQL and NoSQL databases with focus on On-disk and In-memory databases. Section III highlights some existing comparison between these two types of databases. The IoT Database computing scenario that has been setup to evaluate the databases is described in Section IV. Finally, the experiments performed, results and observations are discussed in Section V.

II. Types of Databases

The two major types of databases available on the market today are SQL and NoSQL databases. SQL or Relational databases are the most common database management systems that have been used for a long period of time and relational databases store data in tabular structures linked to each other. The schema of the relational database is defined using the SQL formal language which is further categorised as Data Definition Language (DDL) and Data Manipulation Language (DML). Examples of relational databases are: SQL Server, Oracle Database, Db2, Informix, and MySQL and PostgreSQL [1,2]. On the other hand, NoSQL databases provides a different way of representing data other than tables with relationships. Storing, searching and retrieving data in NoSQL databases are performed using key-value pairs and flexible schemas. The concept of key-value pairs places all the data in a single table and the key represents a unique element. Unlike relational databases where data is structured in rows and columns, NoSQL defines data in a document format that makes it easier to extend without formal updates. Examples of NoSQL databases are: Redis, JanusGraph, MongoDB, Etdc, RabbitMQ and Elasticsearch [2].

978-1-7281-5707-8/20/$31.00 ©2020 IEEE

Both, SQL and NoSQL databases exist as Disk or In-memory database. An In-memory database stores computer data in the main memory (RAM) instead of a disk drive. The disk is mostly used for log and backup purposes after processing and updates are done. When the RAM is closer to the CPU, the amount of time taken to input, process and output data is greatly reduced. In-memory database applications are characterised by rapid response times and real-time data management. Industries that benefit from In-memory databases include telecommunications, banking, travel and gaming. The main considerations for Disk and In-memory databases are [3]:

1) Speed: In-memory databases are faster than traditional databases because they perform less disk input/output operations and therefore, they eliminate the time taken to access data from disk.
2) Volume: Traditional databases depend on the disk drives on which the data is read and written. Different blocks must be read if the data is not in the same location on the hard drive. For In-memory databases, different parts of the database can be managed using direct pointers.
3) Volatility: The main disadvantage of In-memory databases is the volatility of RAM. The data is lost when an In-memory database crashes whereas data can be easily restored from disk databases.

TABLE 1.
DATABASE FEATURE COMPARISON [4,5,6,7,8,9,10,11,12]

	Relational On-disk			Relational In-memory			NoSQL On-disk/In-memory		
Database	**MySQL**	**SQLServer**	**PostgreSQL**	**H2**	**Voltdb**	**HyperSQL**	**Mongodb**	**Couchdb**	**Redis**
Insert	✓	✓	✓	✓	✓	✓	✓	✓	✓
Update	✓	✓	✓	✓	✓	✓	✓	✓	✓
Delete	✓	✓	✓	✓	✓	✓	✓	✓	✓
Select	✓	✓	✓	✓	✓	✓	✓	✓	✓
Index	✓	✓	✓	✓	✓	✓	✓	✓	✓
Group By	✓	✓	✓	✓	✓	✓	✓	✓	✓
Order By	✓	✓	✓	✓	✓	✓	✓	✓	✓
Having	✓	✓	✓	✓	x	✓	✓	✓	✓
Aggregate	✓	✓	✓	✓	✓	✓	✓	✓	✓
Function	✓	✓	✓	✓	✓	✓	✓	✓	x
Subquery	✓	✓	✓	✓	✓	✓	✓	✓	✓
View	✓	✓	✓	✓	✓	✓	✓	✓	x
Cursor	✓	✓	✓	x	✓	✓	✓	x	✓
Stored Procedure	✓	✓	✓	✓	✓	✓	x	x	x
Trigger	✓	✓	✓	✓	x	✓	✓	x	✓

NoSQL databases share many features with relational SQL databases in terms of Create, Read, Update and Delete (CRUD) operations. Irrespective of the type of database, a database management system provide various data management functions that can be classified into four main functional groups: Data definition, Data update, Data retrieval and Data administration. Table 1 above summarises the main features of SQL and NoSQL databases. NoSQL On-disk and In-Memory databases have been combined together since the data is stored in the main memory but persists on disk after each update. The table also highlights three examples of each type of database.

Indexing is a very important concept in databases. Indexes are used to quickly locate data without searching every data record. In other words, indexing helps to locate the exact position of a specific data item in a shorter period of time. Examples of indexes include Primary, Secondary, Clustering, Multi-level and BTree indexes [13]. SQL databases implement indexing mechanisms differently compared to NoSQL databases. In SQL databases, an index is created on one or more columns of a table whereas in NoSQL databases, indexing can be done by associating a key with the location of a corresponding data record.

III. SIMILAR WORKS ON DATABASE COMPARISON

In [14], the authors compared and contrasted the performance of a Disk-based data warehouse and an In-memory database using a real transportation dataset with more than sixty million journeys and around ten million passengers. This dataset consists of passenger journeys and smart cards used in public transportation. Fifteen attributes were chosen from the dataset and the performance of two systems were evaluated using Data Expanding Rate and Running Time. Experiments showed that the In-memory database performed better on most data analytics queries in terms of time, storage and real-time updates. However, some queries executed faster in specific conditions for the Disk-based warehouse since the aggregation in queries are pre-computed whereas in the In-memory database, the aggregations are done in real-time. In [15], three most common document-oriented (NoSQL) databases have been analysed: MongoDB, CouchDB and Couchbase. Testing took into account the execution time of CRUD operations for a single database instance and a distributed environment of two nodes. The findings were then compared with those of the following three relational databases: Microsoft SQL Server, MySQL and PostgreSQL. The comparison of the databases required testing time for all CRUD processes, both in the single instance and in the distributed environment. Testing was done using batch processing for insert, update, select and delete. Each batch of tests was run 50 times for 1000, 10000, and 100000 records respectively. The experiments showed that NoSQL databases performed better than SQL databases. The authors concluded that CouchDB was more suitable for create, update, delete operations and thus, CouchDB would be a good choice for applications that perform intensive write operations. In [16], the authors measured the performance of NoSQL databases in terms of the time taken to complete operations, and the efficient use of RAM during operations. The databases used were: Redis and Memcached as key-value stores, MongoDB as document store, Cassandra as column family, and H2 as In-memory relational database. Memcached gave the best write performance, Redis used the memory more

efficiently than others. The performance of MongoDB decreased considerably when the size of the data increases due to the locking mechanism.

The above databases have been analysed in traditional and distributed computing environments. This work evaluates the performance of SQL and NoSQL databases in an IoT environment which is described in Section IV. Data manipulation includes the CRUD operations together with the use of Indexes, Stored Procedures and Triggers.

IV. IoT Database Setup

The IoT environment used to evaluate the databases is illustrated in Figure 1. The proposed setup consists of an Arduino Uno board to which the following sensors are connected: Temperature/humidity, Moisture and Water level. The sensors are connected to the analog/digital pins of the Arduino board. The Arduino is turn connected to the serial port of a laptop via a usb cable. Data is captured from the sensors and transferred to the databases at regular intervals of 1 second. This setup can be used to monitor plant growth in real-time and based on environmental conditions, alerts can be triggered if the plant growth is not optimal.

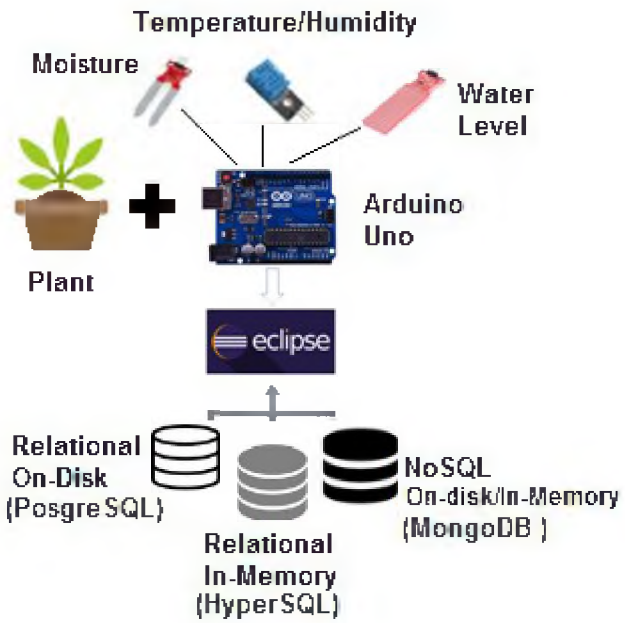

Fig 1. IoT Database Setup

V. Experiments

The experiments performed are analysed and discussed in this section. For each operation, the time taken to execute the operation has been measured in Milliseconds. As the number of records increases, the query time obtained when performing the CRUD operations is relatively less compared to the query time obtained without applying indexing. However, the performance of the three databases differs for each operation. In the following experiments, only the water level parameter has been used in the database operations. Therefore, the analysis of the results is based on one parameter.

A. *Insert Operation*

The results for inserting 100, 500, 1000 and 2000 records are given in Table 2 without indexing and Table 3 with indexing. It can be observed that the average time taken by MongoDB is relatively less than HyperSQL and PostgreSQL. The SQL databases take more time to execute the Insert statement compared to the NoSQL database.

TABLE 2. Insert Operation Without Indexing

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	103266	603491	1103678	2004073
HyperSQL	103265	603485	1103681	2004067
MongoDB	103252	603467	1103683	2004066

TABLE 3. Insert Operation With Indexing

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	103265	603443	1103682	2003896
HyperSQL	103271	603457	1103637	2003970
MongoDB	103217	603456	1103641	2003879

Figure 2 shows the combined results of the Insert operation over time with and without indexing. For all the three databases, the execution time increases with the number of records. However, the difference is not significant and the readings overlap for all the three databases.

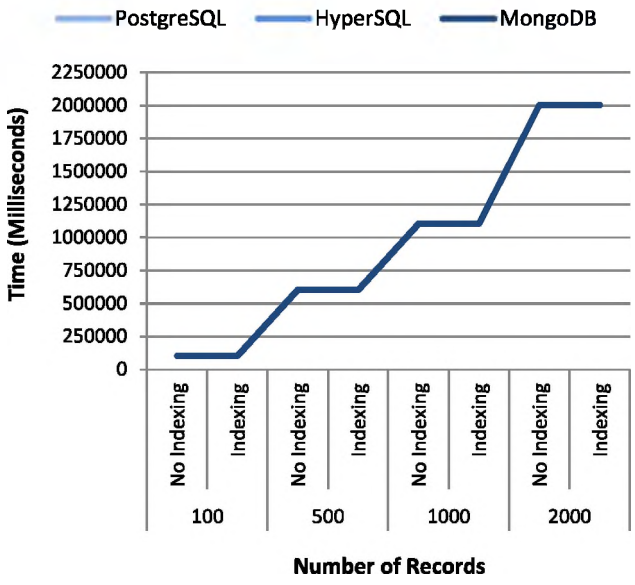

Fig 2. Insert Operation

B. *Select Operation*

The results for selecting 100, 500, 1000 and 2000 records are given in Table 4 without indexing and Table 5 with indexing. In term of data selection, HyperSQL performs better than PostgreSQL on average but the time taken by MongoDB is significantly higher.

TABLE 4. SELECT OPERATION WITHOUT INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	8	95	100	155
HyperSQL	7	36	58	94
MongoDB	196	375	540	698

TABLE 5. SELECT OPERATION WITH INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	8	41	73	76
HyperSQL	11	40	57	69
MongoDB	183	363	498	567

Figure 3 shows the combined results of the Select operation over time with and without indexing. There is a significant difference between the SQL and the NoSQL databases. MongoDB takes much more time to execute the select operation than the SQL databases. Even with indexing, the time taken by MongoDB remains relatively higher as the number of records increases.

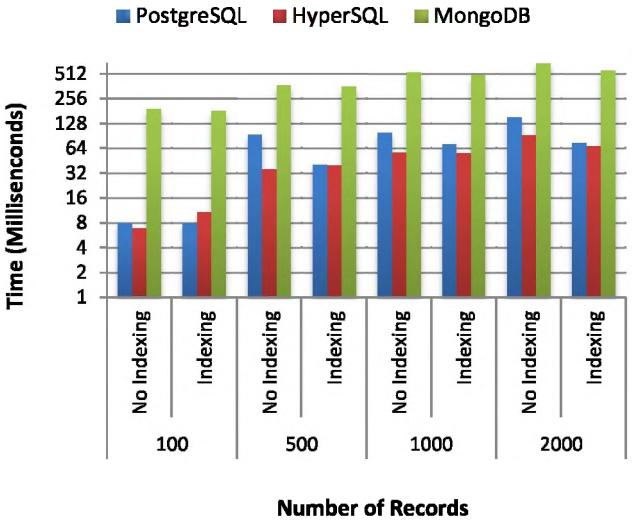

Fig 3. Select Operation

C. *Update Operation*

The results for updating 100, 500, 1000 and 2000 records are given in Table 6 without indexing and Table 7 with indexing. MongoDB takes less time to perform update operations compared to HyperSQL and PostgreSQL.

TABLE 6. UPDATE OPERATION WITHOUT INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	383	553	469	503
HyperSQL	378	340	388	395
MongoDB	202	207	257	275

TABLE 7. UPDATE OPERATION WITH INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	403	461	443	407
HyperSQL	310	282	303	352
MongoDB	216	139	205	212

Figure 4 shows the combined results of the Update operation over time with and without indexing. There is a significant difference between the SQL and the NoSQL databases. MongoDB take much less time to execute the update operation than the SQL databases. With indexing, MongoDB performs even better as the number of records increases while HyperSQL and PostgreSQL have high execution times.

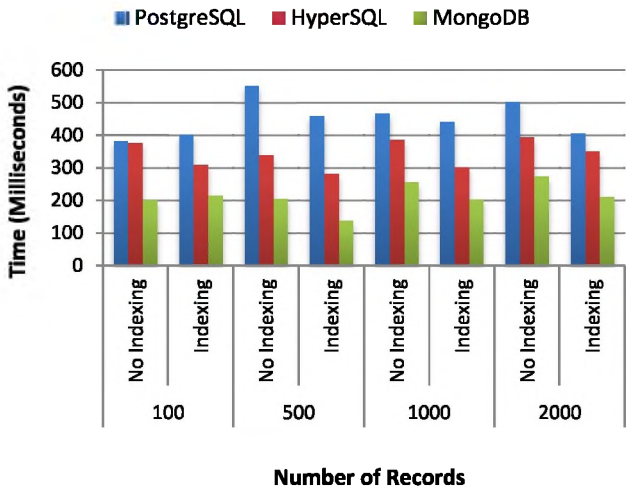

Fig 4. Update Operation

D. *Delete Operation*

The results for deleting 100, 500, 1000 and 2000 records are given in Table 8 without indexing and Table 9 with indexing. It can be observed that MongoDB takes less time to perform the delete operations compared to HyperSQL and PostgreSQL.

TABLE 8. DELETE OPERATION WITHOUT INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	333	422	437	546
HyperSQL	267	271	274	313
MongoDB	250	141	233	309

TABLE 9. DELETE OPERATION WITH INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	331	339	383	307
HyperSQL	266	264	279	287
MongoDB	213	121	240	162

Figure 5 shows the combined results of the Delete operation over time with and without indexing. PostgreSQL has the highest execution time followed by HyperSQL and MongoDB. MongoDB performs well for the deletion of data as the number of records increases. However, the execution time of HyperSQL is less than that of PostgreSQL.

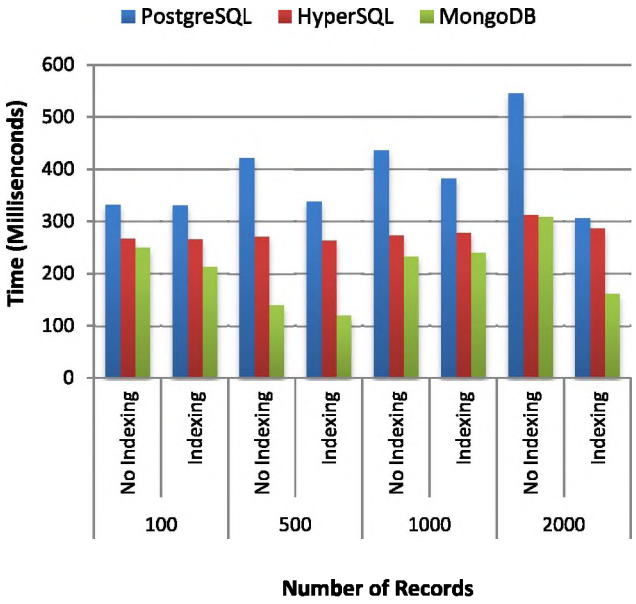

Fig 5. Delete Operation

E. *Stored Procedure*

The results for deleting 100, 500, 1000 and 2000 records using a Stored Procedure are given in Table 10 without indexing and Table 11 with indexing. The Stored Procedure has been applied only for SQL databases using the delete operation. There is a significant difference between the time taken by the On-disk and In-memory database. The time taken by PostgreSQL remained constant as the number of records increased. On the other hand, the time taken by HyperSQL increased with the number of records.

TABLE 10. STORED PROCEDURE (DELETE OPERATION) WITHOUT INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	1	1	1	1
HyperSQL	2760	2924	2859	4987

TABLE 11. STORED PROCEDURE (DELETE OPERATION) WITH INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	1	1	1	1
HyperSQL	3320	4121	3589	6624

Figure 6 shows the combined results of the Stored Procedure over time with and without indexing. The results clearly show that PostgreSQL performs much better with increasing number of records compared to HyperSQL.

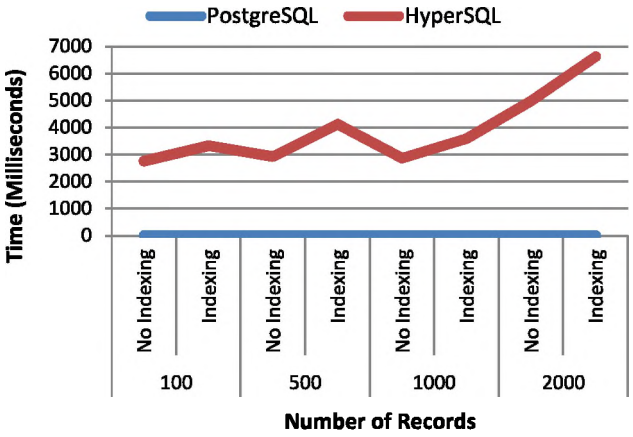

Fig 6. Stored Procedure

F. *Trigger*

The results for inserting 100, 500, 1000 and 2000 records using a Trigger are given in Table 12 without indexing and Table 13 with indexing. Triggers have also been applied only for SQL databases using the Insert operation.

TABLE 12. TRIGGER (INSERT OPERATION) WITHOUT INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	103261	603456	1103641	2003973
HyperSQL	103264	603455	1103616	2003923

TABLE 13. TRIGGER (INSERT OPERATION) WITH INDEXING

Database	Number of Operations / Time (Milliseconds)			
	100	500	1000	2000
PostgreSQL	103266	603491	1103678	2004073
HyperSQL	103265	603485	1103691	2004067

Figure 7 shows the combined results of the trigger over time with and without indexing. For all the three databases, the execution time increases with the number of records. However, the difference is not significant as it can be seen in figure 7 where all the readings from the two databases overlap.

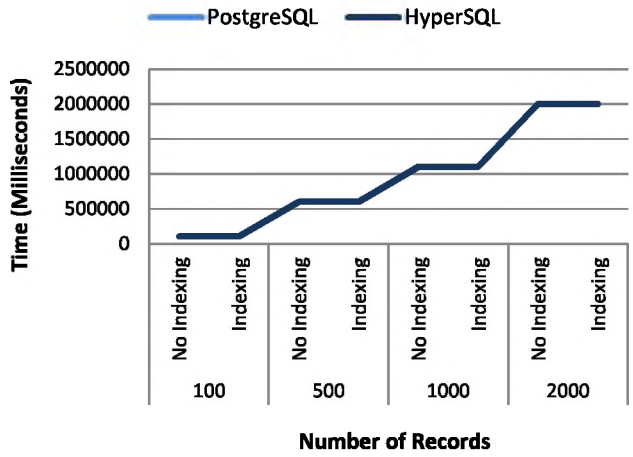

Fig 7. Trigger

CONCLUSION

SQL and NoSQL databases provide different features and cannot replace each other. In this work, the following three types of databases have been compared: PostgreSQL On-disk relational database, HyperSQL In-memory database and MongoDB On-disk/In-memory database. A plant environment has been monitored in real-time using sensor data such as Temperature/Humidy, Moisture and Water levels. Experiments performed using the water level parameter demonstrate there is no single type of database that provides the best performance for all data operations in an IoT environment. For example, Update and Delete operations are faster in NoSQL databases while Select operations execute faster in SQL databases.

As future works, the NoSQL database can further be categorised as On-disk and In-memory. Moreover, in the database experiments performed, only the water level parameter has been considered. As an extension to this work, the execution time can be evaluated in terms of the number of sensor values being manipulated by the database operations. More complex database operations such as nested queries, multiple table joins and views can also be added to evaluate both types of databases.

REFERENCES

[1] Oracle. (2020) What Is a Relational Database? [Online]. https://www.oracle.com/database/what-is-a-relational-database/

[2] IBM Cloud Education. (2019, August) Relational Databases. [Online]. https://www.ibm.com/cloud/learn/relational-databases#toc-examples-o-XT9NJSR1

[3] OmniSci. (2020) In-Memory Database. [Online]. https://www.omnisci.com/technical-glossary/in-memory-database

[4] MySQL. (2020) The Main Features of MySQL. [Online]. https://dev.mysql.com/doc/refman/8.0/en/features.html

[5] Microsoft. (2019, April) Editions and supported features of SQL Server 2019 (15.x). [Online]. https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-version-15?view=sql-server-ver15

[6] Postgresql. (2020, August) Feature Matrix. [Online]. https://www.postgresql.org/about/featurematrix/

[7] H2 Java Database. Features. [Online]. http://www.h2database.com/html/features.html

[8] VoltDB. (2020) Using VoltDB. [Online]. https://docs.voltdb.com/UsingVoltDB/

[9] HyperSQL. (2019, June) HSQLDB - 100% Java Database. [Online]. http://hsqldb.org/web/hsqlFeatures.html

[10] mongoDB. (2020) The database for modern applications. [Online]. https://www.mongodb.com/

[11] CouchDB. (2020) Apache CouchDB® 3.1.0 Documentation. [Online]. https://docs.couchdb.org/en/stable/

[12] Redis. (2020) Redis Documentation. [Online]. https://redis.io/documentation

[13] Tutorialspoint. (2020) DBMS - Indexing. [Online]. https://www.tutorialspoint.com/dbms/dbms_indexing.htm

[14] Pingfu Chao, He Dan, Shazia Sadiq, Kai Zheng, and Xiaofang Zhou, "A performance study on large-scale data analytics using disk-based and in-memory database systems," in *IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2017, pp. 247-254.

[15] Ciprian-Octavian Truica, Florin Radulescu, Alexandru Boicea, and Ion Bucur, "Performance evaluation for CRUD operations in asynchronously replicated document oriented database," Romania, 2018.

[16] Abdullah Talha Kabakus and Resul Karab, "A performance evaluation of in-memory databases," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 4, pp. 520-525, October 2017.