

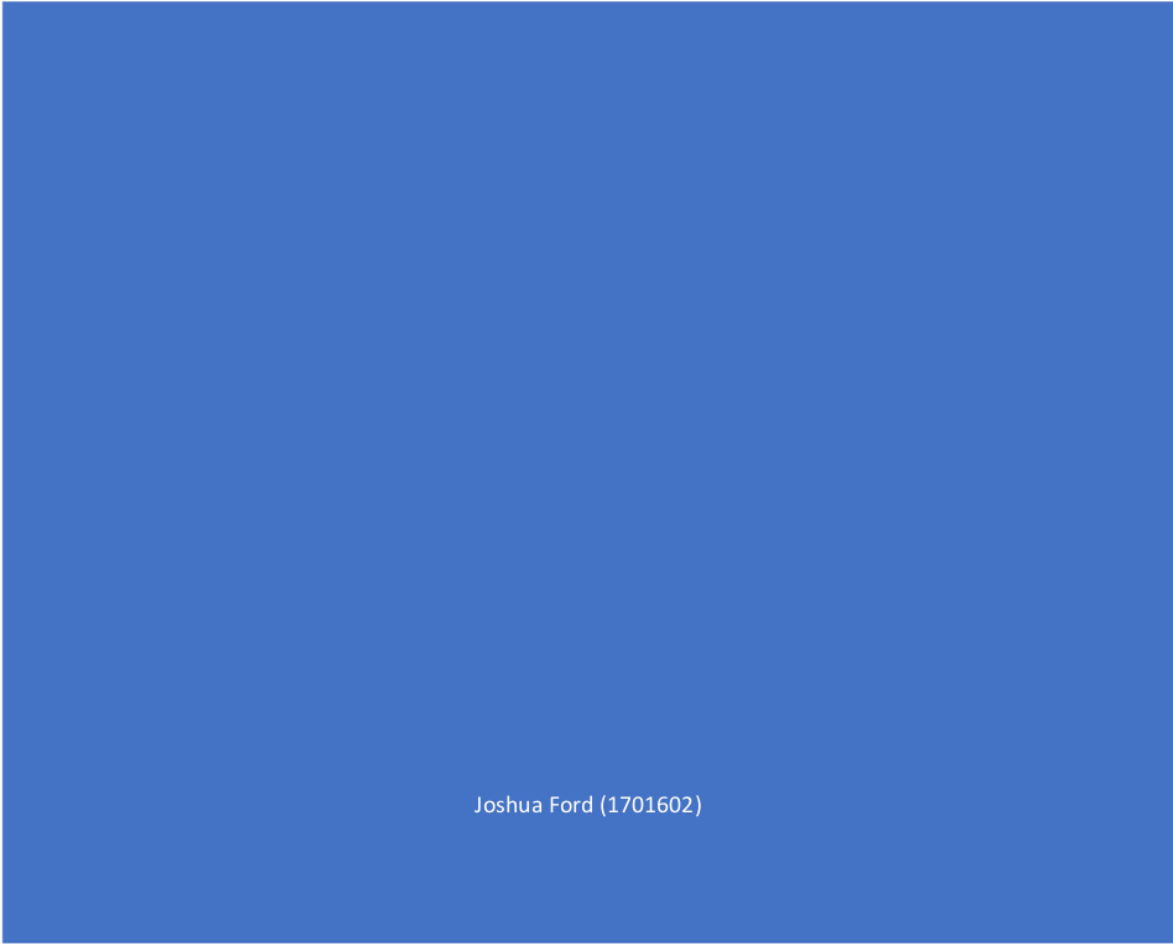
JoshFord1701602

by Joshua Ford

FILE	4098_JOSHUA_FORD_JOSHFORD1701602_226876_1105589534.DOCX (115.45K)		
TIME SUBMITTED	27-MAR-2020 09:05PM (UTC+0000)	WORD COUNT	2645
SUBMISSION ID	122991674	CHARACTER COUNT	13819



BIG DATA ASSIGNMENT



Joshua Ford (1701602)

Contents

1. Introduction	2
2. What is Big Data?	2
3. CAP theorem	2
4. Comparing relational and NoSQL databases	3
5. Chosen Database	4
5.1 Chosen database explanation	4
5.2 Chosen database implementation	5
6. Conclusion	7
7. References	8

1. Introduction

This report will look at what big data is and what it involves, this will mean look into what the term itself means and the four V's that make up big data. Then it will look at the CAP theorem which is consistency, availability and partitioning. Also, it will look at what they are and what compromises are made when creating distributed databases. It will then look at relational and NoSQL databases, how they compare and how different applications are more suited for each type of databases. For the final task, it will look at a chosen NoSQL database and evaluate its use in the scenario explained in the brief. Code will then have to be provided to show the use of the chosen NoSQL database to show the use of CRUD operations.

2. What is Big Data?

Big data is the handling or having use of large amounts of data usually by big companies storing large data sets [1]. It is when the amount of data is so large that different techniques are often used to manage the amount of data that needs to be processed [1]. The data sets used in big data use different techniques because traditional techniques cannot manage that amount of data that the big data techniques can [2].

Big data has four concepts, Volume, velocity, variety and veracity:

- Volume is the size of data being stored, due to lots of types of storage available these days it is usually handled in terabytes and higher [1]. With Big Data handling lots of data, volume is a big part of big data because an example of large amounts of volume in big data would be credit card transactions [3].
- Velocity is how fast the data is transmitted and created, this could be data being added to the database or data being read from the database [4]. High velocity data is data that needs specific processing techniques [3]. Examples of this would be Facebook posts and twitter messages[3], this is because the amount of Facebook posts and twitter messages is so big that they need better ways of retrieving and creating them.
- Variety is the different types of data that is handled [4], the different types of data being structured, semi-structured which is Json and unstructured which is images, audio, video and other similar data types [5]. Variety is a big part of big data because the data that is used in big data is usually from lots of different forms (varieties)[6], this means that variety is present throughout big data because of the different types of data being transmitted.
- Veracity is when the data might not be accurate or correct, meaning data might not be as controllable as accurate and official data [4]. Low veracity data is data that has a lot of useless data[3]. High veracity data would be data that is accurate and very useful[3].

3. CAP theorem

A distributed database which is a database which has many files located in lots of different places or sites [7]. These database' cannot have consistency, availability and partition tolerance all at the same time, there must be a trade-off [8]. Consistency is when the data is consistent across all servers, meaning that if one copy was different it would have low consistency [9]. This is an important to have because without it, one server could have a less up to date copy of something meaning that it would be incorrect. Availability is when something is requested it will always get a response [9]. This is an important factor because if something was requested and no response was sent, the user would not know if it was successful or not. Partition tolerance is when

communications break down in two of the servers, the cluster in the database would continue to process what it was doing even with the communications breakdown[9].

These cannot all be used at once, meaning that each database will not have use of all at once. Examples of these would be:

- Database using consistency and partition tolerance. This would be a database that has sacrificed availability. This would mean that the database would be consistent meaning that it would all client would see the same data, also, it would have partition tolerance meaning that If two servers lost connection or went down, it would continue to work. The sacrifice of availability would mean that there would be no response if something was requested. Also, without it, if something was requested the response would not guarantee the result.
- Database using availability and partition tolerance. This would mean that the database would use availability meaning that the user would get a response from the server when something has been sent to the server rather than waiting for it to be sent across all the servers. Also meaning that it would also get a response from the server for when something else is requested or sent meaning quicker response time. Again, having partition tolerance would mean that if two servers went down or lost connection, it would continue to work.

Each distributed database must make a sacrifice like the examples above. These are sacrifices which each in their own make up for each other because each one makes the database efficient.

4. Comparing relational and NoSQL databases

Relational databases are databases that are stored in tables and can be linked with other tables depending on the data in each table [10]. ACID is a set of properties in a transaction that is used in relational databases [11]. ACID stand for Atomicity, consistency, Isolation, durability. Atomicity is all changes are preformed as a single action and all the changes must be made or none of them are[11]. Consistency is when a transaction is in place, the data is consistent meaning that the data in both ends of the transaction is the same[11]. An example of this would be if a transaction sending important data such as funds and both clients needed to see the funds being sent consistency would be needed so that both users would be able to see the funds. Isolation is when one transaction is being complete and the other transaction cannot begin until the one being completed has finished[11], this is so that the data remains consistent and accurate [12]. Durability is if a transaction is in place and there is a failure somewhere it means that the progress that has been made will not be reversed[11]. A good example of this happening is with a transaction, if a transaction was taking place and sending funds between accounts and a system failure faults the transaction, the durability property will help so that the transaction of sending funds is not undone.

NoSQL databases are used for large amounts of data because of the way that they store data. This is because they do not have a table structure, meaning that the data is being stored differently and not in tables. As stated before, NoSQL databases store a lot of data meaning that when they are being used, they can scale out which means adding more machines to the cluster[13]. The types of NoSQL databases include document based which is MongoDB, column family which is Cassandra and key value which is Redis[13]. Base is Basically available, soft state and eventual consistency. Basically available is when the system is always available, meaning that if one server goes down, it will always be available because it is store across multiple places[13]. This is so that if something is requested and one server is down, it will always be available because there will always be another copy somewhere else. Soft state is the copies of data across the system might be inconsistent even

without input from the user[13]. This means that even without input the state of the system could change over time [13]. Eventual consistency is when a new node is added, data is then copied across all the nodes meaning that there will always be a copy, but there is nothing saying that each copy must have identical copies[13]. This means that the eventual consistency is high but not guaranteed[13].

When comparing to the two, relational and NoSQL, they are both different but can be compared. When wanting to store large amounts of data for a company, NoSQL will be more suitable because the most uses of NoSQL are for companies that store hundreds and thousands worth of data. If the data being store is structured such as lots of columns, relational is more suited[14]. This is because, relational databases have columns and are more suited for data which will have multiple entries, an example would be for a student because it has multiple entries such as name, course age etc. NoSQL would not be suitable because it does not have a table like schema which is not good when storing data like as mentioned[14]. If speed is a necessary, then NoSQL is the best choice because of the ways it can guarantee speeds. It does this by using types like Memcached which stores data that has recently been used for a certain amount of time ready for when it is needed again. This means that when it is queried, it is displayed much quicker than the database looking for it and then displaying it. Relational databases aren't known for speed but more for consistency because if its table schema and more organised layout.

After looking at both types of databases, it shows that both play to their strengths and can be useful for users in their own way and have their own techniques for storing data and then fetching that data in different ways.

5. Chosen Database

5.1 Chosen database explanation

The company stated in the brief needs a suitable database that would be able to hold large data sets and will experience high write and read loads. MongoDB is a database that would be useful in this scenario because it is suited to the requirements of what is needed for the database. MongoDB is good a handling large amounts of data and because it supports an easy use protocol for storing large files [15], aswell as large files MongoDB can both scale up and out, scaling up which is adding more machines which would add performance and scaling out which would add storage which both make it a suitable option for a database which would be able to store large amounts of data [15]. With storing large amounts of data, performance is needed, and MongoDB does not lack in performance. It improves performance by supporting indexing. Indexes help the database by storing small bit of the data's set in an easy to find form [16]. By doing this it helps with faster querying and faster at finding specific data rather than the database manually finding the data. This also goes into high read and write loads. With having high read and write loads being used, indexes can improve this because with adding indexes this will improve the performance and importantly the read speeds. With the company processing a lot of event records from different companies, the different details for each record would have to be stored correctly. This data can be stored as a document, it is a way of storing data using field and value pairs, the data is stored is fields like the field name and value and that would make up a document [17]. It is similar to storing data in a table using SQL, but rather than creating tables like in SQL, documents have fields and there can be lots of different documents than can be searched and queried depending on the field that is being looked for.

5.2 Chosen database implementation

****note:** problem with VM.

MongoDB allows the user to perform CRUD operations which are Create, Read, Update and Delete. The following screenshots show these operations being performed.

The screenshot below shows BSONDocument which is the format, it is then followed by Doc which is the name of the BSONDocument that is going to be created. Followed by Doc and then it is creating a new BSONDocument which will be added into the database. The artist, track and year will be the title and then the values will be the second brackets after the comma. It is then followed by InsertOneAsync(doc), which means insert one Document and then Doc in brackets to specify which document to add.

```
doc = new BSONDocument
{
    { "Artist", "Queen" },
    { "Track", "Bohemian Rhapsody" },
    { "Year", 2018 }
};

await collection.InsertOneAsync(doc);
```

The screenshot below shows the Doc, which has the information about the song with Find, followed by a new BSONDocument then FirstOrDefaultAsync which will display the first document that was in the collection.

```
// Read the first document in the collection
doc = await collection.Find(new BSONDocument()).FirstOrDefaultAsync();
Console.WriteLine("First document in collection is {0}", doc);
```

The screenshot below shows the product in the database being updated, is it specifying the string Artist that has a value of 'Queen'. Then, if the document called doc has no artist called Queen, then display no artist found. If there is an artist found called Queen, then the next field which is Year is specified in the statement that wants to be changed. It says that 'Year' is an INT, that in the BSONDocument with the field Year, followed by what it will be changed to which is +3, which will add 3 onto the year.

```

// Update a single document.
String Artist = "Queen";
doc = await collection.Find(Builders<BsonDocument>.Filter.Eq("Artist", Artist)).FirstOrDefaultAsync();
if (doc == null)
{
    //if db contains no artist called queen, no document found
    Console.WriteLine("No document found for {0}", Artist);
}
else if (doc.Contains("Year"))
{
    //if db does contain artist called queen
    //specify field wanting to be changed, year then add ammount
    doc["Year"] = (int)doc["Year"] + 3;
    Console.WriteLine("[{0}] Changed the year of {1} to {2} in the document", doc["_id"], doc["Artist"], doc["Year"]);
}
else
{
    //check to see if age if in the databse under the artist entered.
    Console.WriteLine("Error: age field did not exist in document");
}

```

This screenshot shows a product being deleted. It is deleting 1 document from the database. It then shows the product being deleted by showing nRemoved: 1.

```

// Delete document
deleted = await collection.DeleteOneAsync(Builders<BsonDocument>.Filter.Eq("i", 100));
if (deleted.IsAcknowledged)
{
    Console.WriteLine("Deleted {0} documents",
        deleted.DeletedCount);
}

```

6. Conclusion

After looking at the different databases and techniques used to maximise the efficiency of database it is found that certain databases have good uses for certain situations. The different type of databases has different schemas and different ways on which they store data and how they are used. Also, after looking at the comparison of the NoSQL and relational databases, its learned that they both worked differently and have different schemas for storing data in using tables and columns and the others not using that techniques but rather focusing on speed and storing large amounts of data sets.

7. References

- [1] J. Grable and A. C. Lyons, 'An Introduction to Big Data.', pp. 17–20, 2018.
- [2] Q. André *et al.*, 'Consumer Choice and Autonomy in the Age of Artificial Intelligence and Big Data', *Cust. Needs Solut.*, vol. 5, no. 1–2, pp. 28–37, Mar. 2018.
- [3] 'The Four V's of Big Data | Big Data Framework©'. [Online]. Available: <https://www.bigdataframework.org/four-vs-of-big-data/>. [Accessed: 19-Mar-2020].
- [4] T. L. Nguyen, 'A Framework for Five Big V's of Big Data and Organizational Culture in Firms', in *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2019, pp. 5411–5413.
- [5] M. Dacey, *Intro to Big Data*. .
- [6] 'Volume, Velocity, Variety: What You Need to Know About Big Data'. [Online]. Available: <https://www.forbes.com/sites/oreillymedia/2012/01/19/volume-velocity-variety-what-you-need-to-know-about-big-data/#626f4e9b1b6d>. [Accessed: 19-Mar-2020].
- [7] 'What is distributed database? - Definition from WhatIs.com'. [Online]. Available: <https://searchoracle.techtarget.com/definition/distributed-database>. [Accessed: 19-Mar-2020].
- [8] F. D. Muñoz-Escóí, R. de Juan-Marín, J.-R. García-Escrivá, J. R. González de Mendivil, and J. M. Bernabéu-Aubán, 'CAP Theorem: Revision of Its Related Consistency Models', *Comput. J.*, vol. 62, no. 6, pp. 943–960, Jun. 2019.
- [9] 'What is the CAP Theorem? | IBM'. [Online]. Available: <https://www.ibm.com/cloud/learn/cap-theorem>. [Accessed: 19-Mar-2020].
- [10] 'Relational Databases Explained | IBM'. [Online]. Available: <https://www.ibm.com/cloud/learn/relational-databases>. [Accessed: 20-Mar-2020].
- [11] 'ACID properties of transactions'. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.4.0/product-overview/acid.html. [Accessed: 20-Mar-2020].
- [12] 'Understanding the Isolation Property in a Database'. [Online]. Available: <https://www.lifewire.com/isolation-definition-1019173>. [Accessed: 20-Mar-2020].
- [13] D. Ganesh Chandra, 'BASE analysis of NoSQL database'. [Online]. Available: https://whel-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=TN_elsevier_sdoi_10_1016_j_future_2015_05_003&context=PC&vid=44WHELF_UWTSD_VU1&lang=en_US&search_scope=CSCOP_EVERYTHING&adaptor=primo_central_multiple_fe&tab=tab1&query=any,contains,base nosql&offset=0. [Accessed: 20-Mar-2020].
- [14] 'Relational vs. NoSQL data | Microsoft Docs'. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data>. [Accessed: 20-Mar-2020].
- [15] K. Chodorow, *MongoDB : the definitive guide*. .
- [16] 'Indexes — MongoDB Manual'. [Online]. Available: <https://docs.mongodb.com/manual/indexes/>. [Accessed: 21-Mar-2020].
- [17] 'Documents — MongoDB Manual'. [Online]. Available:

<https://docs.mongodb.com/manual/core/document/>. [Accessed: 21-Mar-2020].

FINAL GRADE

75/100

GENERAL COMMENTS

Instructor

a (20): 18

b (20): 12

c (20): 15

di (20): 18

dii (20): 12

Disucssion:

I wanted an example that demonstrated the CAP theorem.

SQL vs NoSQL applications were a bit vague in parts. (See notes)

MongoDB chosen and reasoning was good, except you could have analysed the structure of the data required i.e. JSON might suggest semi-structured data. Code was little simplistic but demo'd some use of MongoDB.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

Not necessarily, it always takes time for changes to propogate through a cluster



Comment 2

Cassandra has columns



Comment 3

This is a little vague.

What do we mean by "speed"?



Comment 4

NoSQL databases are consistent otherwise they would always give the wrong results

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10
