

Big Data Assignment

by Runa Bibi

FILE	4054_RUNA_BIBI_BIG_DATA_ASSIGNMENT_226876_111282213.DOCX (378.98K)		
TIME SUBMITTED	27-MAR-2020 11:51PM (UTC+0000)	WORD COUNT	2849
SUBMISSION ID	123002010	CHARACTER COUNT	16581

Big Data

Big Data Assignment

Mike Dacey

Runa Bibi (1702463)

27/03/2020

Table of Contents

1. Introduction	2
2. Big Data	2
3. Consistency, Availability and Partitioning (CAP) Theorem	3
4. Relational and NoSQL Databases.....	4
5. Zang Technologies	6
6. References.....	8

Table of Figures

Figure 1 - MetLife The Wall	4
Figure 2 - Create a document	7
Figure 3 - Reading documents in database.....	7
Figure 4 - Updating a single document.....	7
Figure 5 - Deleting a single document.....	7
Figure 6 - Finding a specific document.....	7

1. Introduction

This report will include a discussion on Big Data and the Consistency, Availability and Partitioning (CAP) Theorem. It will also compare NoSQL and relational databases and relevant applications to each. A NoSQL database will be chosen and evaluated for the company Zang Technologies with code demonstrating CRUD operations on this selected NoSQL database.

2. Big Data

Big Data is large volumes of data that cannot be processed by conventional technologies available in changing degrees of complexities and produced at varying velocities [1]. It has characteristics referred to as the 4 V's which are: volume, veracity, variety and velocity [2]. Volume is in relation to the vast quantity of data collated by the company which can grow up to terabytes and even petabytes [3]. Veracity is the reliability of data and how accurate it is [4]. Variety refers to the different forms of data i.e. structured, semi-structured and unstructured [5].

Structured data is that which is easily stored and organised in a recognisable way e.g. a relational database that makes it simple to query [6]. Semi-structured data is a version of structured data that does not follow a structure like a relational database but has tags which can separate similar elements and implement field and record hierarchies and does not need a schema definition [7]. Unstructured data has no certain structure and includes items such as bitmap objects/images, email, text, video etc. [8]. Velocity is the speed of the transfer rate of data from the source to the destination [9].

Due to these characteristics, big data has huge value in business. For example, the higher the volume of data on customers, the better the understanding of them which helps businesses make better decisions regarding customer relationship management (CRM) [10]. The more varied the data that is collected, the more "multi-faceted" [10] of an assessment is built on the customers [10] which helps businesses be more conscious of customer needs [10]. The faster data is injected and analysed, the quicker businesses can make the right decisions to meet their objectives for CRM [10]. Also, for the right decisions to be made, data needs to be consistent, consolidated, current and cleansed and cannot be assumed to be accurate based on the volume alone [11].

3. Consistency, Availability and Partitioning (CAP)

Theorem

A distributed database is a group of databases that are kept at different sites and appear as one database to the user [12]. Each database can be handled by a different architectures and different database management systems that allocate the implementation of transactions [13]. The CAP theorem, developed by Eric Brewer [14], refers to Consistency, Availability and Partitioning and how distributed databases can only have two out of three of these properties simultaneously [15].

Consistency in distributed databases ensures a transaction is atomic and the system either fails or remains consistent [16]. Users should have the same view of the data at any time and show the latest data when queried [17]. For data to be consistent, when it is written to one node, it has to be copied to every other node before the write is effective [18]. Availability in the system means any request made by the client needs to eventually obtain a response [19]. Every read and write request should return a response from all non-failing nodes in a “reasonable amount of time” [20]. Partition tolerance is that any system failure or data loss does not disrupt the system and it continues to operate [21].

One option for software engineers to take is a Consistency/Partition Tolerance system where a timeout error can occur while waiting for a response from the partitioned node [22]. An error can also be returned by the system dependant on the scenario [22]. Choosing Consistency over Availability should be done when atomic read and writes are part of your business requirements [22]. An example of a Consistency/Partition Tolerance program is MongoDB, a distributed document-oriented database program [23]. MetLife, an insurance provider, uses MongoDB for their customer service application “The Wall”, designed to emulate Facebook [24]. It collates data from over 70 legacy systems into a single record and runs in two data centres across six servers, storing around 24 terabytes of data [24]. Availability can be compromised, as data consistency and accuracy is more important in this scenario e.g. for customer transactions and policy details [24].

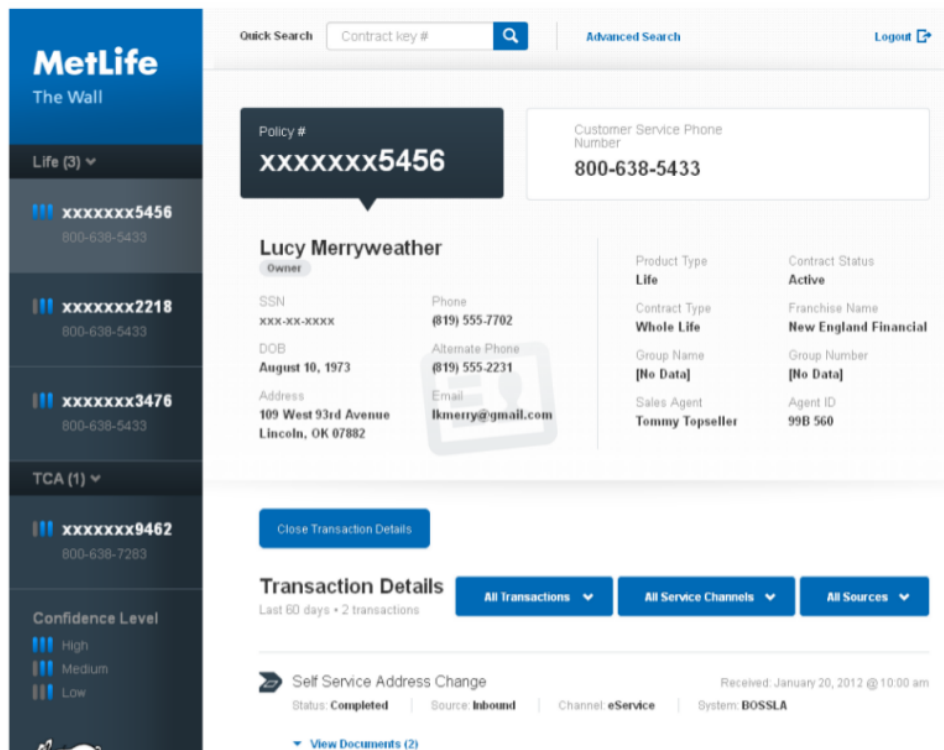


Figure 1 - MetLife The Wall

4. Relational and NoSQL Databases

Relational databases are largely based on ideas from Edgar Codd's 1970 paper wherein he discussed a data model based on relationships [25]. Relational databases, in their simplest form, represent data in "two-dimensional tables" [26] which each contain columns known as attributes and rows known as records with unique identifiers or primary keys [27]. These tables are joined through foreign keys [28].

Transactions in relational databases should share the properties referred to as ACID: Atomicity, Consistency, Isolation and Durability [29]. Atomicity is where a transaction needs to be fulfilled to completion and the state of the database should be based on before the transaction execution or after it has been executed or failed [30].

Consistency is where data needs to be consistent from the start of a transaction to the end [31]. Transactions need to be independent of other transactions that are happening at the same time which gives them the isolation property [32]. Changes

made by the transaction that have been committed should stay regardless if there is a hardware or software failure i.e. durability [33].

There are many suitable applications which play to the strengths of relational databases. Data consistency as provided by relational databases, is vital for applications where the user needs to always see the system in a consistent state; such as an ATM deposit where they check their account, they would need to see it updated to the new amount [34]. Data is also easy to query [35] which makes it suitable for keeping track of inventory e.g. stock in a shop where you can query the availability of a product. Another strength of relational databases is the security they provide so companies can control access permissions [36] depending on each employees' position such as HR having access to employee's salary information but regular employee's not having this access of other employee's salaries.

NoSQL or "not only SQL" databases were developed due to its high scalability compared to relational databases [37]. It was designed to deal with large sets of data and does not follow any relational schema [38]. BASE in NoSQL is Basically Available, Soft state and Eventual Consistency [39]. These characteristics are needed to allow for scalability and flexibility [40]. Basically available means data will be available for querying regardless of the state of the data i.e. if it is in the stale state [41]. Soft state is where the system state can change over time with no extra input [42]. Eventual consistency means when the system stops getting input it will be inconsistent for an acceptable amount of time and then eventually become consistent [43].

There are several types of NoSQL databases e.g. Key-Value Pair Databases, Document Databases, Graph Databases and Column Family Databases [44]. One huge advantage of NoSQL is its ability to handle big data [45] so one useful application for it is analyses for social trends and tracking of huge unstructured data [46]. Another example of an application for NoSQL is detecting fraud in the e-commerce and banking sector [47]. Real-time fraud detection faces several challenges e.g. the large volumes of data that needs to be processed, the increasing complexity of fraud detection models and growing demands of customers [47].

Graph databases can help solve these problems because complex traversals can be accomplished from one record to another quickly compared to relational databases [47]. This is due to the fact it has no tables separating the data so for example, if a customer is purchasing from a new business, the two objects are linked by “a new edge element” which makes for a very fast traversal [47].

A further application for NoSQL is in Internet of Things (IoT) which is a network infrastructure that is dynamic and global with “self-configuring capabilities” [48] and interoperable and standard communication protocols where virtual and physical “things” have physical attributes, virtual personalities and identities, are joined in the information network and “use intelligent interfaces” [48]. NoSQL databases can provide the scalability, flexibility and agility required for the management of more assorted data which has been created by millions and millions of devices, sensors and gateways, each containing their own data structures and possibly becoming linked over numerous years [49].

NoSQL databases, more specifically MongoDB, can be used for a product catalogue, and to search for product availability [50]. MongoDB is non-relational which aids in the flexibility of the catalogue, so each document does not need to follow the same schema [50]. Therefore, each product only needs attributes that fit that product [50]. Several queries can be performed on the product catalogue e.g. sorting albums by the year they were produced or by a certain word in the title [50].

5. Zang Technologies

MongoDB has been chosen for the Zang Technologies scenario. It has been chosen because the records can be simply stored inside a collection as separate documents [51]. Data can also be represented in JSON and BSON (Binary JSON) as is needed by Zang Technologies and is almost as powerful as SQL regarding its queries [51]. MongoDB can also handle a variety of data types and its flexibility means the database schema can change with the business requirements [52].

Here is some code displaying CRUD operations on the database:

```
// Create a document
doc = new BsonDocument
{
    { "name", "Bob Smith" },
    { "age", 42 },
    { "email", "bobsmith@gmail.com" },
    { "phone number", 07248294273 }
};
```

Figure 2 - Create a document

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

// Read all the documents in the collection
Console.WriteLine("Reading all the documents");
```

Figure 3 - Reading documents in database

```
// Update a single document

String name = "Bob Smith";
UpdateResult updated;
updated = await collection.UpdateOneAsync(Builders<BsonDocument>.Filter.Eq("_id", doc["_id"]),
    Builders<BsonDocument>.Update.Inc("age", 1)); // Adds one to the age
doc = await collection.Find(Builders<BsonDocument>.Filter.Eq("name", name)).FirstOrDefaultAsync();
Console.WriteLine("Changed document now contains {0}\n", doc);
```

Figure 4 - Updating a single document

```
DeleteResult deleted;

// Delete one document
deleted = await collection.DeleteOneAsync(Builders<BsonDocument>.Filter.Eq("i", 110));
if (deleted.IsAcknowledged)
{
    Console.WriteLine("Deleted {0} documents",
        deleted.DeletedCount);
}
```

Figure 5 - Deleting a single document

```
// Find a specific document
doc = await collection.Find(Builders<BsonDocument>.Filter.Eq("i", 68)).FirstOrDefaultAsync();
Console.WriteLine("Finding document with i=68 {0}", doc);
```

Figure 6 - Finding a specific document

6. References

- [1] K. Krishnan, "Defining Big Data," in *Data Warehousing in the Age of Big Data*, Waltham, Elsevier Inc., 2013, p. 5.
- [2] G. Kapil, A. Agrawal and R. A. Khan, "A study of big data characteristics," in *International Conference on Communication and Electronics Systems*, Lucknow, 2016.
- [3] H. J. Hadi, A. H. Shnain, S. Hadishaheed and A. H. Ahmad, "BIG DATA AND FIVE V'S CHARACTERISTICS," *International Journal of Advances in Electronics and Computer Science*, vol. 2, pp. 16-23, 2015.
- [4] J. Williamson, "The Four V's of Big Data," in *Getting a Big Data Job For Dummies*, Hoboken, John Wiley & Sons, Inc., 2015, pp. 29-31.
- [5] R. Kitchin and G. McArdle, "What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets," *Big Data & Society*, vol. 3, pp. 1-10, 2016.
- [6] S. Praveen and U. Chandra, "Influence of Structured, SemiStructured, Unstructured data on," *International Journal of Scientific & Engineering Research*, vol. 8, no. 12, pp. 67-69, 2017.
- [7] O. Rusu, I. Halcu, O. Grigoriu, G. Neculoiu, V. Sandulescu, M. Marinescu and V. Marinescu, "Converting unstructured and semi-structured data into knowledge," in *RoEduNet IEEE International Conference*, Bucharest, 2013.
- [8] A. C. Eberundu, "Unstructured Data: an overview of the data of Big Data," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 38, pp. 46-50, 2016.
- [9] A. S. Panimalar, V. S. Shree and V. A. Kathrine, "The 17 V's Of Big Data," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 9, pp. 329-333, 2017.

- [10] B. Rossi, "How to measure the value of big data," InformationAge, 21 August 2015. [Online]. Available: <https://www.information-age.com/how-measure-value-big-data-123460041/>. [Accessed 8 January 2020].
- [11] jKool, "The Four Essential V's for a Big Data Analytics Platform," 3 July 2017. [Online]. Available: <https://www.jkoolcloud.com/four-essential-vs-big-data-analytics-platform/>. [Accessed 8 January 2020].
- [12] K. K. Ezéchiél, S. Kant and R. Agarwal, "A systematic review on Distributed Databases Systems and their techniques," *Journal of Theoretical and Applied Information Technology*, vol. 96, pp. 236-266, 2019.
- [13] S. Gupta, K. Saroha and Bhawna, "Fundamental Research of Distributed Database," *International Journal of Computer Science and Management Studies*, vol. 11, no. 2, pp. 138-146, 2011.
- [14] S. Srinivasa and V. Bhatnagar, "CAP Theorem and Its Origin," in *Big Data Analytics*, Heidelberg, Springer, 2012, pp. 9-10.
- [15] A. Fowler, "Availability Approaches," in *NoSQL For Dummies*, Hoboken, John Wiley & Sons, Inc., 2015, pp. 45-47.
- [16] S. Sakr and M. M. Gaber, "The CAP Theorem and Beyond," in *Large Scale and Big Data: Processing and Management*, Boca Raton, Taylor & Francis Group, LLC, 2014, pp. 328-330.
- [17] T. John and P. Misra, "CAP Theorem," in *Data Lake for Enterprises*, Birmingham, Packt Publishing, 2017, pp. 75-76.
- [18] IBM Cloud Education, "CAP Theorem," IBM, 14 November 2019. [Online]. Available: <https://www.ibm.com/cloud/learn/cap-theorem>. [Accessed 8 January 2020].
- [19] S. Gilbert and N. A. Lynch, "Perspectives on the CAP Theorem," *Computer*, vol. 45, no. 2, pp. 30-36, 2012.

- [20] A. Mehra, "Understanding the CAP Theorem," Database Zone, 30 April 2019. [Online]. Available: <https://dzone.com/articles/understanding-the-cap-theorem>. [Accessed 8 January 2020].
- [21] J. Celko, "NoSQL and Transaction Processing," in *Joe Celko's Complete Guide to NoSQL*, Burlington, Morgan Kaufmann Publishers, 2014, pp. 1-14.
- [22] R. Greiner, "CAP Theorem: Revisited," 14 August 2014. [Online]. Available: <https://robertgreiner.com/cap-theorem-revisited/>. [Accessed 9 January 2020].
- [23] MongoDB, "The database for modern applications," [Online]. Available: <https://www.mongodb.com/>. [Accessed 9 January 2020].
- [24] S. Gopinath, "Real World Use Cases of MongoDB," 22 May 2019. [Online]. Available: <https://www.edureka.co/blog/real-world-use-cases-of-mongodb/>. [Accessed 9 January 2020].
- [25] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377-387, 1970.
- [26] C. Ritchie, "The relational database," in *Relational Database Principles*, London, Continuum, 2002, pp. 8-13.
- [27] M. Rouse, "relational database," TechTarget, [Online]. Available: <https://searchdatamanagement.techtarget.com/definition/relational-database>. [Accessed 10 January 2020].
- [28] J. A. Kreibich, "Database Design," in *Using SQLite: Small. Fast. Reliable. Choose Any Three.*, Sebastapol, O'Reilly Media, Inc., 2010, pp. 87-115.
- [29] Ian, "What does ACID mean in Database Systems?," 20 June 2016. [Online]. Available: <https://database.guide/what-is-acid-in-databases/>. [Accessed 10 January 2020].
- [30] TutorialsPoint, "DBMS - Transaction," [Online]. Available: https://www.tutorialspoint.com/dbms/dbms_transaction.htm. [Accessed 10 January 2020].

- [31] IBM, "ACID properties of transactions," [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.4.0/product-overview/acid.html. [Accessed 10 January 2020].
- [32] FairCom, "Database Talk: What is ACID compliance?," 26 November 2018. [Online]. Available: <https://www.faircom.com/insights/database-talk-acid-compliance>. [Accessed 10 January 2020].
- [33] Educative, "What are ACID properties in a database?," [Online]. Available: <https://www.educative.io/edpresso/what-are-acid-properties-in-a-database>. [Accessed 10 January 2020].
- [34] Oracle UK, "What a Relational Database Is," [Online]. Available: <https://www.oracle.com/uk/database/what-is-a-relational-database/>. [Accessed 10 January 2020].
- [35] Guru99, "Relational Data Model in DBMS: Concepts, Constraints, Example," [Online]. Available: <https://www.guru99.com/relational-data-model-dbms.html>. [Accessed 10 January 2020].
- [36] Techspirited, "Advantages of Relational Databases," [Online]. Available: <https://techspirited.com/advantages-of-relational-databases>. [Accessed 10 January 2020].
- [37] A. Nayak, A. Poriya and D. Poojary, "Type of NOSQL Databases and its Comparison with Relational Databases," *International Journal of Applied Information Systems*, vol. 5, no. 4, pp. 16-19, 2013.
- [38] M. Rouse, "NoSQL (Not Only SQL database)," TechTarget, [Online]. Available: <https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL>. [Accessed 11 January 2020].
- [39] M. Dave, "SQL and NoSQL Databases," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 8, pp. 20-27, 2012.

- [40] D. deRoos, P. C. Zikopoulos, B. Brown, R. Coss and R. B. Melnyk, "Hadoop and the Data Warehouse: Friends or Foes?," in *Hadoop for Dummies*, Hoboken, John Wiley & Sons, Inc., 2014, pp. 157-179.
- [41] V. Das, "Introduction to NoSQL," in *Learning Redis*, Birmingham, Packt Publishing, 2015, pp. 1-21.
- [42] G. Vaish, "What SQL is and what it is not," in *Getting Started with Nosql*, Birmingham, Packt Publishing, 2013, pp. 5-7.
- [43] J. Celko, "1.6 Base," in *Joe Celko's Complete Guide to NoSQL: What Every SQL Professional Needs to Know about Non-Relational Databases*, Waltham, Elsevier Inc., 2014, p. 11.
- [44] D. Sullivan, *NoSQL for Mere Mortals*, Ann Arbor: Pearson Education, Inc., 2015.
- [45] Guru99, "NoSQL Tutorial: Learn NoSQL Features, Types, What is, Advantages," [Online]. Available: <https://www.guru99.com/nosql-tutorial.html>. [Accessed 11 January 2020].
- [46] R. Barooah, "Applications That Work Best With NoSQL Database," Clarion Technologies, [Online]. Available: <https://www.clariontech.com/blog/applications-that-work-best-with-nosql-database>. [Accessed 11 January 2020].
- [47] A. Mahmoud, "Detecting complex fraud in real time with Graph databases," IBM , 19 January 2017. [Online]. Available: <https://developer.ibm.com/dwblog/2017/detecting-complex-fraud-real-time-graph-databases/>. [Accessed 11 January 2020].
- [48] O. Vermesan and P. Friess, "Putting the Internet of Things Forward to the Next Level," in *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*, Denmark, River Publishers, 2014, pp. 3-6.

- [49] M. Asay, "Why You Need NoSQL For The Internet Of Things," ReadWrite, 28 November 2014. [Online]. Available: <https://readwrite.com/2014/11/28/internet-of-things-nosql-data/>. [Accessed 12 January 2020].
- [50] MongoDB, "Product Catalog," [Online]. Available: <https://docs.mongodb.com/ecosystem/use-cases/product-catalog/>. [Accessed 12 January 2020].
- [51] P. Jayaram, "When to Use (and Not to Use) MongoDB," DZone, 30 November 2016. [Online]. Available: <https://dzone.com/articles/why-mongodb>. [Accessed 12 January 2020].
- [52] MongoDB, "What are the main differences between MongoDB and MySQL?," [Online]. Available: <https://www.mongodb.com/compare/mongodb-mysql>. [Accessed 12 January 2020].

Big Data Assignment

GRADEMARK REPORT

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

a (20): 18

b (20): 14

c (20): 17

di (20): 9

dii (20): 12

Disucssion:

Discussion of CAP theorem could have been a little more detailed but you have provided a reasonable example.

SQL vs NoSQL discussion is good but you have also provided good examples of applications for both.

You should have provided more evaluation of the requirements of Zang Technologies and the appropriateness of MongoDB.

Code provided does demo CRUD operations although it is a bit simplistic.

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14
