

Big Data Report 2 - Liam Salter - 1702256

by Liam Salter

FILE	2963_LIAM_SALTER_BIG_DATA_REPORT_2_-_LIAM_SALTER_-_1702256_245230_246569255.DOCX (136.76K)		
TIME SUBMITTED	09-MAY-2020 10:48PM (UTC+0100)	WORD COUNT	2601
SUBMISSION ID	126858869	CHARACTER COUNT	14321



BIG DATA.

Semester 2.



APRIL 16, 2020

NAME: LIAM SALTER | STUDENT NUMBER: 1702256.
COURSE: WEB DEVELOPMENT.

Table of Contents

1. Introduction.....	2
2. Key-Value Databases.....	2
2.1. Benefits.....	2
2.2. Drawbacks.....	2
2.3. Key Features.....	3
2.4. Suitable Example Application.....	3
2.5. Potential SQL Requirements.....	3
2.6. Database Design.....	3
3. Document-Oriented Databases.....	3
3.1. Benefits.....	3
3.2. Drawbacks.....	4
3.3. Terminology Meanings.....	5
3.4. Suitable Example Application.....	5
3.5. Database Design.....	5
4. Graph Databases.....	5
4.1. Benefits.....	5
4.2. Drawbacks.....	6
4.3. Summary.....	6
5. Conclusion.....	6
6. Appendices.....	7
6.1. Appendix 1.....	7
6.2. Appendix 2.....	7
6.3. Appendix 3.....	8
References	9

1. Introduction.

In this report, varying types of databases will be discussed. This will be done by detailing the benefits, drawbacks, key features, providing an example of application, discussing potential SQL requirements, and providing a database designs for the Key-Value Database. The benefits, drawbacks, key features, terminology meanings, providing an example of application and database design for Document Oriented Databases will be detailed. Finally, the benefits, drawbacks and key features of Graph Databases will be detailed.

2. Key-Value Databases.

2.1. Benefits.

Key-Value databases have a number of benefits. One of the benefits of using a key-value database is that data of any type can be stored within the database, such as images, videos, text files, numbers, words etc. This means that key-value databases can be very flexible in purpose.

Another major benefit to using a key value database is that they are uniquely indexed from the key field. This means that values from the database can be found and retrieved extremely fast, even if the database contains a small or very large amount of records.

Another major benefit to using key-value databases is that they typically utilise less memory to store a database, compared to other databases, which means the database gains drastic performance increases compared to other types of databases.

Another major benefit to key-value databases is that data is treated like a collection, which then contain different fields. This again adds to the flexibility of the database [1].

2.2. Drawbacks.

However, key-value databases do have some drawbacks. One of the drawbacks to using a key-value database is that only simpler queries are the fastest and most efficient. When more complicated queries are operated on the database, efficiency and speed are likely to be impacted.

Another major drawback to using key-value databases is that it is extremely difficult to perform simple SQL operations on the database, such as grouping or joining. This would require a database redesign and restructuring and would likely impact the querying efficiency and speed of the key-value database, one of its major benefits.

Another drawback to using key-value databases is that a very limited constraint functionality is available to use. Typical SQL constraints such as foreign keys and not null are not available to use. This means that collections cannot relate to each other, and fields which should have required information within them can be left empty. Some NoSQL managements program offer some limited a structured query language.

Another drawback to key-value databases is that if value types need to be converted to another type, this has performance impacts. For example, for key-value databases to be in the form of a relational model, all values must be of the same type. This means that if dates are stored as strings, the strings will need to be converted to a date format before they can be joined to other tables. Which, in turn, has performance costs [2].

2.3. Key Features.

Key-value database are able to store a vast amount of data. And as previously mentioned, a key-value database can store any type of data, such as images, videos, text, numbers, word documents etc.

Another feature of key-value databases is key choice. In key-value databases, there are very little constraints when it comes to how data is stored and arranged. However, one constraint is that keys MUST be unique in any given namespace (also known as “buckets”). For example, the namespace “phones” can have to key “name”. This means there cannot be another key named “name” in the “phones” namespace. However, there could be a separate namespace known as “employees” with the key “name”. This is allowed data can be queried through the namespace.

Another feature of key-value databases is the use of hash functions to create the generation of keys and namespaces. Hash functions in key-value databases allow the conversion of long strings, numbers, objects etc. into hash values, which in turn, can be used as a key. However, rarely, hash values may not always end up being unique. When two or more different values produce the same hash value, this is known as a “collision”. To avoid collisions and prevent it from happening, a collision resolution strategy must be employed. There are two primary strategies for collision resolution, these being Open Addressing and Chaining [3].

2.4. Suitable Example Application.

The suitable example that was chosen was an online shopping store. As seen in example of the key-value datasets, the key-value pairs are utilising a few of the benefits of key-value databases. One of these benefits is the scalability. As the namespaces are unique, the number of accounts can be scaled outwards and the number of baskets stored can be scaled outwards. Another benefit of key-value databases that was utilised was that keys can have the same name; however, they cannot be within the same namespace. Another advantage of key-value databases that was utilised was that any form of data can be input into the database, such as text and numbers. However, an image of the product could also be put into the basket namespace and it would still work as intended.

2.5. Potential SQL Requirements.

However, with the particular design of the key value database and the scale, it could be substituted with a traditional relational database design. As seen in the image displaying the relational database equivalent, account can be stored in a “User Accounts” table and baskets can be stored in a “Baskets” table. Also, the user id of accounts can be a foreign key in the baskets table.

2.6. Database Design.

Please see Appendix 1 for the design of the Key-Value Database. Please see Appendix 2 for the SQL database extension.

3. Document-Oriented Databases.

3.1. Benefits.

As with Key-Value databases, Document-Oriented databases also have a number of benefits. One of the benefits to using a document-oriented database is that they are very flexible. The structure of any single document does not have to be consistent. This means that even large quantities of unstructured data can be stored in the database.

Another major benefit of using a document-oriented database is that the process of bringing in new data is very simple. The new data must only be included in a small number of datasets, compared to a typical relational database, where the new data criteria must be added to all datasets.

Another major benefit to using document-oriented databases is that instead of storing data across a vast number of linked tables, all the data is instead stored in one place. This means that there is a large speed and/or performance boost to querying the database. However, the performance boost is only applicable where there are no relational elements implemented onto the database. Introducing interlinking into document-oriented database increase the complexity of the database and therefore the performance boost is lost.

Another major benefit to document-oriented databases is that they are highly scalable. Document-oriented database are easily scalable as collections of documents grow. This means that there is reduced maintenance on the database as it continues to grow and the cost of scaling heavily reduced [4].

3.2. Drawbacks.

As with any other form of database, document-oriented databases also have some drawbacks. One of the drawbacks of using a document-oriented database is that database design problems can occur. As document-oriented databases are very flexible by nature, this can encourage developers to become lazy with the design and cause poor design decisions. An example of this could be collections containing documents too dissimilar to the entities the current document contains. Collections should only contain documents with entities which relate to each other.

Another major drawback of using a document-oriented database is that due to denormalization, there is a far higher probability of data being duplicated more than a handful of times. In turn, this increases the chance of data anomalies occurring and causes unnecessary space to be taken up and used in the database.

Another flaw of using a document-oriented database is that, as with any form of NoSQL database, there is absolutely no native support for any structured query languages, such as SQL. As previously mentioned, some NoSQL management systems do have a very limited support for structured query languages, however the majority of the database structuring will have to be designed and built by the developer(s).

Another major drawback of using a document-oriented database relates to the overly flexible nature of the database. One problem that can arise from overly different documents is that performance can take a hit. This is due to how the different documents are stored on the disk. However, this can be slightly mitigated through the use of indexing. Indexing will slightly help with the speed of the querying; however, indexing is typically reserved for queries that are executed frequently and is bad practice to index hardly used queries.

Another problem that can arise from overly different documents is that a substantial amount of overly complicated code can be produced. This is due to the large amount of conditional functions required to handle the content of different types of documents [5].

3.3. Terminology Meanings.

Simply, the term “document” is the definition of the unit of storing data within a document-oriented database. A document does not have a pre-defined set of constraints, instead it is up to the developer(s) decide how to document should be structured. Documents contain a set of key-value pairs. Documents also contain both structure information and data. For the most part, documents use JSON as a template style for storing data.

Secondly, the term “collection” is the definition in which how and where documents are stored. A collection can store a numerous number of documents. Simply, a collection is a set of documents. The general purpose of a collection is to group similar documents which pertain to the to the same type of entity. Collections in document-oriented databases are also primarily used to perform CRUD operations. Also, collections are designed with scalability in mind [6].

3.4. Suitable Example Application.

The suitable example application that was chosen was a user account section on a website. One of the main benefits that was used was that new data can easily be introduced. If the user decided to update a setting, the document could easily be changed and updated. Another benefit that was used is that all the user data is collated in one location, the “Users” collection. Another benefit that could be used on this implementation is that it is scalable, as when a new user signs up, a new user document is created and added to the Users collection. Also, the documents were demoralised to increase efficiency. Over demoralisation was also avoided.

3.5. Database Design.

Please see Appendix 3 for the design of the Key-Value Database.

4. Graph Databases.

4.1. Benefits.

Like the previous forms of NoSQL, graph databases also come with a numerous amount of benefits. However, there are three primary benefits in which why a social media would want to use a graph database over any other NoSQL database. The first of these benefits is performance. For ultimate data relationship handling, performance improvement provided from graph databases is extremely high. Graph data performance will stay constant, reliable level, even as the amount of data that needs to be handled increases. Comparatively, with traditional relational databases, as the number of queries increase and/or the depth of relationships increase, database performance will take a substantial hit.

The second major benefit a social media company using a graph database would have is flexibility. With graph databases, development teams are able to keep up with the business demands as the structure and schema of a graph model database is malleable and can change as the business goals change. Essentially, rather than spending a large amount of time modelling a domain ahead of time, developers are able to append to the pre existing graph structure database without causing problems with the existing functionality.

The third and final major benefit of a social media company using a graph database would have is agility. As many development teams across the industry is moving towards an agile, test-driven development work practices. Graph databases align perfectly with this methodology of development. Current standards of graph databases are fit for frictionless development and graceful systems maintenance [7].

4.2. Drawbacks.

Again, as with the other forms of NoSQL databases, graph databases come with a few drawbacks. One of these drawbacks being that even though scalability is possible with a graph database, it is a task in of itself. As graph databases are, by nature, architecturally one-tier, growth, or scalability presents a mathematical challenge.

Another disadvantage of using a graph database, like all other NoSQL databases, there is absolutely no native support for a structured query language. This means that any forms of structuring must be designed and implemented by hand, by the developer(s). However, as previously mentioned, some NoSQL managements systems provide a limited support for structured query language [8].

4.3. Summary.

In summary, it would be an extremely good idea if a social media company chose to utilise a graph database as graph databases utilises the strength of data connections. From the ground-up, a graph database is designed to handle well connected data. In the current climate of everything being data driven, all data increasing in volume and connectedness, utilising graph databases would be the perfect for a social media company, as social media has become extremely data driven.

5. Conclusion.

In conclusion, key-value databases have been discussed by detailing the benefits and drawbacks of using a key-value database, the key features of a key-value database. A suitable example application of a key-value database was created and displayed and potential reasoning why an SQL database would have to be implemented onto said database was also detailed and displayed. Document-oriented databases were also discussed by detailing the benefits and drawbacks of using a document-oriented database and terminology behind document-oriented databases. A suitable example application of document-oriented database was created and displayed. Finally, graph-based databases were discussed by detailing the benefits and drawbacks of using a graph database, for the purpose of why a social media company would want to utilise said database.

6. Appendices.

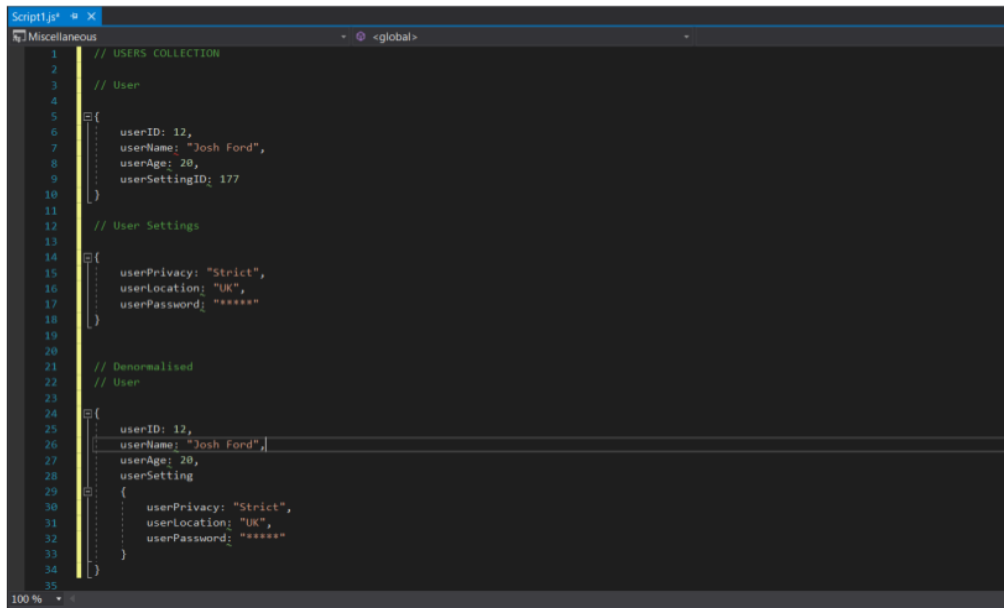
6.1. Appendix 1.

Josh Account 1		Josh Basket 1	
"name"	"josh ford"	"item1"	"book"
"age"	20	"item2"	"iphone"
"address"	"64 dev lane"	"price"	550.5
"id"	1234	"id"	1234
Dan Account 1		Dan Basket 1	
"name"	"daniel newman"	"item1"	"socks"
"age"	20	"item2"	"aws sub"
"address"	"1 dev lane"	"price"	100000
"id"	1235	"id"	1235

6.2. Appendix 2.

Accounts			
name	age	address	id
josh ford	20	64 dev lane	1234
daniel newman	20	1 dev lane	1235
Baskets			
items	price	account id	
book, iphone	550.5	1234	
socks, aws sub	100000	1235	

6.3. Appendix 3.



```
1 // USERS COLLECTION
2
3 // User
4
5 {
6   userID: 12,
7   userName: "Josh Ford",
8   userAge: 20,
9   userSettingID: 177
10 }
11
12 // User Settings
13
14 {
15   userPrivacy: "Strict",
16   userLocation: "UK",
17   userPassword: "*****"
18 }
19
20
21 // Denormalised
22 // User
23
24 {
25   userID: 12,
26   userName: "Josh Ford",
27   userAge: 20,
28   userSetting:
29     {
30       userPrivacy: "Strict",
31       userLocation: "UK",
32       userPassword: "*****"
33     }
34 }
35
```

References

- [1] BI-Insider, "Key-Value NoSQL Databases," BI-Insider, 14 January 2019. [Online]. Available: <https://bi-insider.com/posts/key-value-nosql-databases/>. [Accessed 9 May 2020].
- [2] L. Olivera, "Everything you need to know about NoSQL databases," DEV, 5 June 2019. [Online]. Available: <https://dev.to/lmolivera/everything-you-need-to-know-about-nosql-databases-3o3h#dis>. [Accessed 9 May 2020].
- [3] S. S. Abhishek T, "What is Collision and what are different collision resolution techniques.," Ques10, January 2020. [Online]. Available: <https://www.ques10.com/p/35013/what-is-collision-and-what-are-different-collision/>. [Accessed 9 May 2020].
- [4] 1&1, "Document Databases: How Do Document Stores Work?," IONOS by 1&1, 23 March 2020. [Online]. Available: <https://www.ionos.com/digitalguide/hosting/technical-matters/document-database/>. [Accessed 9 May 2020].
- [5] M. Dacey, *3a. Intro To Document Databases*, Swansea: Michael Dacey | UWTSO, 2020.
- [6] W3Resource, "Databases, documents and collections in MongoDB," W3Resource, 26 February 2020. [Online]. Available: <https://www.w3resource.com/mongodb/databases-documents-collections.php>. [Accessed 9 May 2020].
- [7] Neo4J, "Why Graph Databases?," Neo4J, [Online]. Available: <https://neo4j.com/why-graph-databases/>. [Accessed 9 May 2020].
- [8] 1&1, "Graph databases explained," IONOS by 1&1, 7 November 2019. [Online]. Available: <https://www.ionos.co.uk/digitalguide/hosting/technical-matters/graph-database/>. [Accessed 9 May 2020].

Big Data Report 2 - Liam Salter - 1702256

GRADEMARK REPORT

FINAL GRADE

57 / 100

GENERAL COMMENTS

Instructor

KV DB (10): 10
KV Des (20): 12

Doc DB (20): 14
Doc Des (20): 11

Graph DB (20): 10

Feedback

=====

The database design for the KV DB demonstrated some of the advantages of KV.

Doc DB discussion contains a few misunderstandings. The design is just a short paragraph with a little "code", more analysis of the design was expected.

For the graph DB you have covered a number of interesting points but the debate of the applicability of this database was a bit limited. What about linking friends together, etc.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

No Doc DB do have their own query language. There is no **standard** language, such as SQL, but they do allow similar CRUD operations.



Comment 2

This is comparative, compared with SQL these types of NoSQL database are simple inorder to remain fast.



Comment 3

Perhaps a better way of putting this would be that the user has write more code because the database does much less than SQL databases.

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10
