

Wanzi Su BD Assignment 2

by Wanzi Su

FILE	17848_WANZI_SU_WANZI_SU_BD_ASSIGNMENT_2_245230_1711649323.DOCX (240.97K)		
TIME SUBMITTED	11-MAY-2020 05:13PM (UTC+0100)	WORD COUNT	4355
SUBMISSION ID	127104089	CHARACTER COUNT	22156

Contents

0. Tables and Figures	1
1. Introduction	1
2. Key-Value Database	2
3. Document-oriented database	5
4. Graph Database.....	10
5. Conclusion	12
6. Reference	13

0. Tables and Figures

Table 1 : Key-Value Database for User Management System.....	4
Table 2 : SQL Database for User Management System	4
Table 3 : Document Database for Q&A System	9
Table 4 : Denormalisation	10
Figure 1 : Data Model of MongoDB [5]	6
Figure 2 : Structure of Graph Database [9]	11

1. Introduction

According to Cristian Andrei BARON (2016), the NoSQL databases can be divided based on the optimization strategy and on the different kinds of tasks that they resolve in four groups: key-value, document-based, wide column and graph. [1] This essay describes the benefits, drawbacks, and the features of three types of NoSQL database: key-value database, document database, and graph database. Another work is providing example applications and example database designs for these databases.

The essay can be divided into three parts. The first part is about the key-value database and a User Management System application using both the key-value database and SQL database. Including why this application needs to use a key-value database and why SQL database.

The second part is about the document database and a Q&A System application. Including why document database has so many benefits.

The third part is about the graph database. Including why the social media site would want to adopt this database.

These three NoSQL databases respectively have their own characteristics. Understanding them would allow you to have a general overview and a general comparison of them in future work.

2. Key-Value Database

Key-Value database is one type of NoSQL database and as its name has shown, the data that it stores contains a key and a value. Therefore, its structure is pretty simple, which makes the query really easy. On the one hand, every key should be different from each other because the key is the identifier of this data. Since the key is unique, all the queries would only use the key to operate the data. This is one of the reasons why the queries are fast and reliable. On the other hand, the value can be any type of data, which means you do not have to consider the type of data. String, hash, list, set or sorted set. Whatever type the data is, just store it.

Overall, we can conclude the benefits and the drawbacks of the key-value database. The first benefit is that key-value database is really simple. It only contains keys and values. The queries are easy and the value does not have a limitation or classification on the data type because it is schemaless.

The second benefit is that the queries are very fast, which also allows more queries to be processed at the same time. Some key-value database would store value in the cache, which is even faster. For example, Memcached database stores value in the cache and Redis database can choose to store value in the cache too. Here, W. Puangsaijai and S. Puntheeranurak (2017) did a lot of experiments on the operation time comparing Redis key-value database and MariaDB SQL database and the conclusion was: *Redis has better runtime performance of insert operation, delete operation, update operation especially updating by using specific conditions, and select operation in querying single result of the simple query as well as querying data of complex query. Because Redis has many advantages such as flexible, no fixed schema, and easy to use. [2]*

The third benefit is that key-value database has good scalability. It has no fixed schema so it can add as much data as you wish. One thing to mention is that although the database has no schema, the hardware has a storage limit. For example, the cache would remove the old data if it is full. Sainan Zhu (2015) gave a similar view: *In the key-value database, any new data item, no matter what the structure is, can be put into the store at any time without affecting existing data items and the availability of the database system. [3]*

The fourth benefit is that the cost of the key-value database is very low. It is simple so the data is easy to store and the high speed allows more queries to be processed.

On the other side, there are also many drawbacks. First, you can only use a key to find the value because all queries are only for keys. This makes it impossible to search value and return the key. Second, there is no fuzzy query for values. The queries should be pointing at the keys which must be unique. Third, there is no way to partly modify the value, which is bad for complex data. For example, if the value is "Tom, 1705154, Math, 65", you cannot only change "65" and you have to type in the whole value again. Fourth, the key-value database usually does not have CRUD operations. Its queries are very basic like SET or GET.

To solve the problem of no value search and no fuzzy query, there are several solutions. The keys are normally unique identifiers but they can also be some information related to the value. In this case, when we search keys, we can also kind of search their values. Or alternatively, they can contain both. This is called "namespace". For example, for the data of two students, the keys could be "StudentA:id:001" and "StudentB:id:002". "StudentA", "StudentB", and "id" are the namespaces and the number is the identifier for "id". In this case, we can search "Student*" to get the data of those two students. Another good solution is to directly convert the value into an identifier. For instance, hash functions can convert a value "Hello" into a hash value "12", which can be used as the identifier. We can also use both a namespace and a hash value as the key.

Here is a User Management System application well suited for the key-value database. We can use the key-value database to store the username and the password. The function provided here is to allow the users to create a new account using a username and a password, log into the system with this account or change the password. All these operations would only affect the key-value database. In common cases, the users would not want the username and password of their account to be very long or complex. Simple and short username as well as password can be easier for the users to remember, even though the security is not good. Considering this, the application would limit the length of the username and the password but also have complexity

requirement on the password, where schemaless is necessary. As a result, the username and the password would be short, which meets both the requirements of the users and the simplicity of the key-value database. The key-value database design is in table 1.

Key	Value
username:001	password:SSSsss123&
username:002	password:D*c45R
username:003	password:WD54^d
username:004	password:SS212fg
username:005	password:?sw55seW

Table 1: Key-Value Database for User Management System

SQL database is also used in this application to store the other detailed information of the user like the address, the telephone number, the email address, and the job title then also store the username as the key. The users themselves can read their information and edit their information, which has nothing to do with the key-value database. The application should provide a function for the users to search any piece of the detailed information to find another user or other users. Table 2 is the SQL database design.

Username	Address	Tel	Email	Job
001	Oxford St, Swansea	12345678911	001@gmail.com	Student
002	Western St, Swansea	12345678912	002@gmail.com	Software Developer
003	Westbury St, Swansea	12345678913	003@gmail.com	Java Systems Architect
004	Walter Rd, Swansea	12345678914	004@gmail.com	Web Developer
005	Oxford St, Swansea	12345678915	005@gmail.com	Student

Table 2: SQL Database for User Management System

On the one hand, the users would use the username and the password frequently because they need to log in before they can get the services. In addition to that, the password would sometimes need to be changed because of many reasons like forgetting the password or the password disclosure. Therefore, the username and the password require a fast query in the key-value database.

The key-value database has a low cost and a high performance so it would allow massive users to log in or create an account at the same time under less resource pressure. Moreover, scalability is also one of the reasons why the key-value database can process a large number of account

creations. Therefore, the key-value database allows this application to handle queries from more users.

On the other hand, the detailed information would not be accessed so frequently, which means fast queries are not so necessary. Furthermore, the data is large and complex so it is not suitable to be stored in the key-value database. If we store this detailed information in the key-value database, we have to rewrite all the information every time we want to change a piece of information, which would make the key-value database even slower than SQL database. Key-value database also does not allow the users to search a telephone number or email address to find a user unless it can be contained as a namespace and an identifier in the key. Then the key would be even longer than the value and this would be terrible to the performance, which is not what we would expect.

The key-value database does not have a schema but SQL database has, which means we can make restrictions to the format of each column in SQL database. For example, the telephone number must be number. This is one more thing that the key-value database cannot do.

3. Document-oriented database

Document-oriented database (for short, document database) is another type of NoSQL database. According to B. Vela et al. (2017), *Document databases store documents, which can be grouped into collections of similar documents. [4]* We can learn from this literature that the most important thing to understand document database is the explanation of two terms “document” and “collection”.

Here, S. kumari (2018) provided his description of collection and document then a straightforward model figure to illustrate his idea: *A collection contains many documents as shown in data model of figure 1. Each document contains data in the form of key-value pair. Collections can be indexed, which improves lookup and sorting performance. [5]*



Figure 1: Data Model of MongoDB [5]

Therefore, document is basically a group of key-value pair. We can consider the document database as an evolution of the key-value database. Cristian Andrei BARON (2016) held a similar opinion: *The document-based model represents the next iteration of key-value. [1]* The difference between the data in the key-value database and the document is that the document has a structure. If the order of the key-value pairs is different, the documents are different even if those key-value pairs are completely the same. However, the document is not totally structured. Just like key-value database, the document database has no schema.

As B. Vela et al. (2017) and S. kumari (2018) all agreed, collection is a group of documents. These document must have something in common otherwise there is no meaning of them grouping together. This view is supported by E. W. Vaandering (2003), who wrote: *Each document had "topics" (for instance a type of physics or a detector) that were user-defined. [6]* Normally, we can use collections to represent different types of objects. The documents within the same collection should belong to one type and have similar attributes. For example, if a document is representing a black dog and another document is representing a yellow dog, they can be grouped into a collection representing dogs. These documents should have some common attributes like the colour, age, or weight.

One thing to mention is that the design of collection allows easier queries to operate the documents and the collections, which means you can directly add a document, delete a document, add a collection, or delete a collection. This is the superiority of the document database compared to the key-value database: it provides some overall operations.

Another difference between the document database and the key-value database is that both the key and the value are searchable in the key-value pairs of the document. According to Cristian Andrei BARON (2016), *in the same time both keys and values remain fully searchable. [1]*

We can summary from the discussion above that the document database is mostly like the key-value database so it has some common benefits and drawbacks as the key-value database. However, in some aspects, it is more like a SQL database, for instance, the structure of the documents. This is enough for us to list the benefits and the drawbacks of the document database. J. Scott (2016) proved this view: *As NoSQL databases have cemented their place in the industry, document databases have become a dominant choice due to their flexibility to support use cases where the key-value databases are not a perfect fit. These document databases have also shown that they can support many use cases where key-value databases excel.* [7]

One of the benefits is that the document database is still simple and has no schema. You could say that it is some collections of groups of key-value pairs so it is essentially key-value pairs only more complex. The queries are slightly more complex than key-value database but much simpler than SQL database.

Another benefit is high performance. The design of collections and documents allows the users to operate a group of data with only one query. Just like key-value database, it would use cache to store the data, whose process speed is higher than most memories. Kristina (2013) described a document database MongoDB in her book, which stated: *It uses as much of RAM as it can as its cache.* [8]

Scalability is also a benefit of the document database. You can add collections and documents as long as you want, which would make the database scale out. This view is also supported by Kristina (2013), who wrote: *MongoDB was designed to scale out. Its document-oriented data model makes it easier for it to split up data across multiple servers.* [8]

The last benefit is flexibility. The data type has no limitation and the database would not store any empty value because it has no schema. In addition to that, the document database has more kinds of queries than the key-value database, which enables more operation. Cristian Andrei BARON (2016) agreed that: *Unlike the simpler version of key-value stores, in this type of store, the value column contains semi-structured data, mainly pairs of attribute names and values. The value of a column can reach hundreds of attribute pairs, where the type and the number of attributes can vary from one row to other.* [1]

The drawbacks are quite clear. First, there is no rule for which documents should be grouped into a collection so things would happen when documents belonging to different type are grouped

into the same collection. This would make the queries divided for different documents and waste both the time on directing to the right document and the resource of implementing the queries. Second, the queries are still not as standard as the SQL language.

To illustrate the benefits above, a Q&A System application is provided. This application is designed to allow the users to use their account to log in and ask questions or answer the questions. The users can also delete or update their own questions. On the one hand, the document database would record the users' operation in Event collection including the time and whether this operation is a success. The delete and update operations have one more attribute, which is the target question id. Besides the target question id, the create operation has one more attribute, which is the type of the message created, question or answer. On the other hand, the questions and the answers are also stored in the Q&A collection of the database including the message and the source that this message is from. The answers should have an additional attribute to point toward its question. Therefore, except login and logout, these two collections would respond to other three operations from the users. The login and logout only modify the Event collection. Table 3 is the document database design.

Collection	Document	Key-Value Pair
Event	Login	name:Tom
		time:7am
		state:success
	Create1	name:Tom
		time:8am
		state:failed
		questionId:1
		type:question
	Delete	name:Tom
		time:8am
		state:success
		questionId:1
	Create2	name:Tom
		time:8am

		state:success
		questionId:2
		type:question
	Create3	name:Bob
		time:9am
		state:success
		questionId:2
		type:answer
Q&A	Qusetion1	questionId:2
		message:Where is Sam?
		from:Tom
	Answer1	questionId:1  1
		answerId:1
		message:Sam took your pen.
		from:Bob
	Answer2	questionId:2
		answerId:1  2
		message:At home.
		from:Bob

Table 3: Document Database for Q&A System

In the application, on the one hand, the operation and the data type are both various. Document database provides enough kinds of queries and has no data type restriction. On the other hand, the structure of the data is not complex and only contains events or messages. Events contain operations and operations contain attributes, which is the same logic as messages. The simplicity of the document database can apply to this.

When the users are in this system, the application would record their every operation including the login, the create, the update, the delete, and the logout. This is a huge task and the records are numerous if there are many users operating. In this case, the high performance and the

scalability of the document database can play an important role. The instant answer also requires fast speed. You would not want to wait for others' answer delayed by poor performance. In fact, there is another way to increase even more performance, which is denormalisation.

Collection	Document	Key-Value Pair	
Event	Create3	name:Bob	
		time:9am	
		state:success	
		question	questionId:2
			message:Where is Sam?
			from:Tom
		type:answer	

Table 4: Denormalisation

Table 4 is the Create3 document after denormalisation. In this case, when you want to check which question the answer is to and what its content is, you do not need to find the questionId in Event collection and then use this id to search the Q&A collection. You can see all the related information in one collection, which makes the query speed up to a certain extent. It is worth noting that denormalisation may cause more redundancy and inconsistency, which can be seen in table 4 that the document is even more complex.

4. Graph Database

To understand what is graph database, we must know what is graph first. Here, the graph refers to the graph concept in graph theory. X. Tao et al. (2018) proposed a similar view: *Graph database is a type of NoSQL database, whose data storage structure and query method are based on the graph theory. [9]*

According to the graph theory research of Abdul Majeed and Ibtisam Rauf (2020), a *graph is a set of nodes and edges. The nodes are referred to as vertices/vertex, and the edges are lines or arcs that connect any two/multiple nodes in a graph. [10]* Therefore, there are two basic things of the graph: nodes and edges. Then the question is: what are nodes and edges representing in the graph database?

It is not very difficult to find the answer that the nodes are the entities and the edges are the relations. This is because we need at least two entities to build a relation, which is the same as the nodes and the edges. This conclusion is supported by O. Orel et al. (2016), who provided more specific information: *In general, data entities and relations among them are referred to as nodes and edges. In some cases, both nodes and edges can have properties attached to them, and edges can be directed from starting node to ending node. [11]*

In the graph database, the edges do not have properties but the nodes would be labelled, which means the graph database usually has four elements: the nodes representing the entities, the edges representing the relations, the properties attached to nodes and edges, labels for each node. Therefore, for the social media site, the graph database is indeed appropriate for storing the users in the nodes and storing relationships in the edges. Then, we can store some additional information as the properties or the labels.

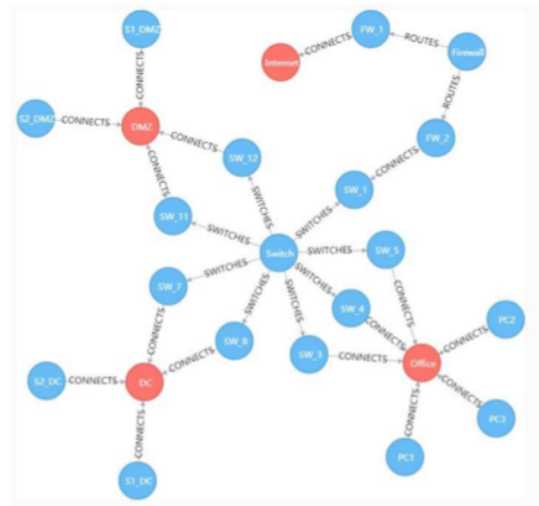


Figure 2: Structure of Graph Database [9]

Figure 2 is basically a structure of the graph database provided by X. Tao et al. (2018). As we can see in figure 2, the structure of the graph database is very different from other databases. In other databases, the data would normally be found by searching from a key to another key in a specific order. However, in the graph database, the search would go from a node through an edge to another node, which is more like the social relationship in reality. In other words, other databases search would be more like a roll call in reality. As a result, the graph database is easier for the users or developers to understand and this would also more likely to meet the reality.

The scalability is also one of the graph database's benefits. You can feel free to add a node, a property or a label to any nodes. Every node is a start node so the database can scale out in any

direction. The more nodes the graph database has, the better its scalability is. Therefore, when facing a huge amount of data like the massive users on social media site in this scenario, the scalability of the graph database is the best choice.

The graph database can provide excellent relational query and high performance, which are really helpful to process so much data. The structure of the graph database has determined its advantage on the relational query. The graph database can provide the algorithm in graph theory so it does not have to iterate through all the nodes. It can take a shortcut to find the target node, which makes it perform better no matter for a big task or a small task. Bowen Kan et al. (2017) made a comparison among the graph database and other databases: *Relational databases are highly inefficient at describing and traversing complex connections and relationships... Most NoSQL databases, whether key-value-, document-, or column-oriented, store sets of disconnected documents/values/columns. This makes it difficult to use them for connected data graphs. NoSQL is suitable for simple structured big data instead of data with plentiful properties and labels... Graph database is useful for power grids because of the continuously increasing scale of the power grid, extremely wide distribution of power station and high-performance requirement of power grid needed. [12]*

One thing to mention is that the graph database can provide all the CRUD operations and it has no data type limitation just like other NoSQL databases so the nodes can be very flexible. The users can add any type of data and delete a node if they want. This flexibility can give web users more freedom and provide a better experience.

In conclusion, the graph database is considered to be really useful to the social media site in this scenario. It is easy to understand, more realistic, easy to scale out, good at the relational query, performing well, and really flexible.

5. Conclusion

In part 2, I explained the structure of the key-value database. Then I discussed the benefits and the drawbacks of the key-value database. It has some benefits on simplicity, performance, scalability and cost but it also does not have value search, fuzzy query, partial modification, or CRUD operations. In the end, I used a User Management System application and designed two database examples for it to demonstrate my views.

In part 3, I described the terms “document” and “collection” in the document database. Next, I included its advantages and disadvantages. It is simple and very fast. Schemaless makes it flexible and performance allows it to scale out. However, a wrong document might be added to the collection and the queries are not standard. A Q&A System application and its database example are provided to illustrate those benefits then I explained a little about “denormalisation”.

In part 4, I talked about the structure of the graph database, which is very different from other databases. Then I gave many reasons why the graph database is useful in the scenario.

6. Reference

- [1]. Cristian Andrei BARON, “NoSQL Key-Value DBs Riak and Redis”, Database Systems Journal. 2016;VI(4):3-10
- [2]. W. Puangsaijai and S. Puntheeranurak, “A comparative study of relational database and key-value database for big data applications”, 2017 International Electrical Engineering Congress (iEECON), Pattaya, 2017, pp. 1-4, doi: 10.1109/IEECON.2017.8075813.
- [3]. Sainan Zhu, “Creating a NoSQL database for the Internet of Things: Creating a key-value store on the SensibleThings platform”, 2015
- [4]. B. Vela, J. M. Caverio, P. Cáceres, A. Sierra and C. E. Cuesta, “Defining a NoSQL Document Database of Accessible Transport Routes”, 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, 2017, pp. 1125-1129, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.172.
- [5]. S. kumari, “IMPLEMENTATION OF MAP-REDUCE PARADIGM IN MONGODB AND COUCHDB”, 2018, International Journal of Advanced Research in Computer Science, vol. 9, no. 2, pp. 724-729.
- [6]. E. W. Vaandering, “A Web Based Document Database”, 2003
- [7]. J. Scott, “Streamlining Business With a NoSQL Document Database”, 2016, Big Data Quarterly, vol. 2, no. 1, pp. 33.
- [8]. Kristina Chodorow, “MongoDB: The Definitive Guide”, 2013, Second edition. edn. O'Reilly Media.
- [9]. X. Tao, Y. Liu, F. Zhao et al., “Graph database-based network security situation awareness data storage method”. J Wireless Com Network 2018, 294 (2018).
- [10]. Abdul Majeed and Ibtisam Rauf, “Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks”, 2020, Inventions

[11]. O. Orel, Slaven Zakošek and M. Baranovič, "Property Oriented Relational-To-Graph Database Conversion", *Automatika*, vol. 57, (3), 2016.

[12]. Bowen Kan, Wendong Zhu, Guangyi Liu, Xi Chen, Di Shi and Weiqing Yu, "Topology Modeling and Analysis of a Power Grid Network Using a Graph Database", 2017, *International Journal of Computational Intelligence Systems*

Wanzi Su BD Assignment 2

GRADEMARK REPORT

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

KV DB (10): 10
KV Des (20): 15

Doc DB (20): 18
Doc Des (20): 15

Graph DB (20): 12

Feedback

=====

I would have liked to have seen a slightly more complex example of KV DB but the analysis was quite good.

The Doc DB design was a little hard to follow, but the basic ideas were present. Table 3 didn't seem right, and I'm not sure about "user" which I assume is the field used for identifying the document.

For the graph DB you have covered a number of interesting points but the debate of the applicability of this database was limited.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9



Comment 1

2?



Comment 2

2?

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14
