MongoDB to Simplify Your Database Structure

Elvina Riama K. Situmorang (13514045)
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514045@stei.itb.ac.id, vina9196@gmail.com

***Abstract*—Currently, the development of technology is growing rapidly. The impact of this situation is our world become uncertain. The uncertainty world need a flexibility of technology especially in database. There relational database (RDBMS) that already known before 20th century. But, RDBMS is too rigid to follow the growth of technology. So that in 21st century there are non-relational (NoSQL). One of the famous database system is MongoDB. This paper will give a case that show how the MongoDB will simplify the database structure.**

***Keywords*—flexibility, RDBMS, NoSQL, MongoDB, simplify.**

I. Introduction

In database management system (DBMS) there are 4 types of structural database, such as hierarchical database, network database, relational database, and object-oriented database[1]. Nowadays, there is unstructured database. The unstructured database usually called as NoSQL or non-relational database. It's called as unstructured database because the database has a flexible schema. Actually, the database itself has structure, the structure is adopted from JSON's structure or syntax.

There are a lot examples of unstructured database type. One of widely known types is MongoDB. The MongoDB itself was invented around November 2009[2]. The aim of NoSQL is to simplify the design of database and the process of scaling. By definition, NoSQL is the next generation of database that provide non-relational database, easily scaling especially in scalable horizontal database, distributed database, and an open source database[3].

The NoSQL was born to answer the need of the industry. Nowadays, if we build some applications, there are some X-factors that we can't define in the beginning. That's why somehow we need a storage of the data in a database system which has high flexibility. The NoSQL also respond to our dynamic world. Recently, our world especially in the technology area is growing rapidly. This development impact to various factors, include the database. Many of applications become more complicated to survive because of the impact of development itself.

II. Literature Study

A. Database Management System

In a database management system (DBMS) there are two main components, a collection of data that have relation to each other that usually called as database and a sequence of program to store, access, modify the data from database. The main purpose of DBMS is to get the required information effectively and efficiently[4]. The system in DBMS must consider about the security and how to keep the integrity of the database. Beside that, the system also have to notice if there is a crash and how to prevent it from happening too often, so the data will keep safe. So basicly DBMS is a software that has responsibility of managing database. In this paper, we will focus on two types data models of DBMS, the relational database model usually represented by SQL language and non-relational database model usually called as NoSQL.

There are several examples of SQL such as MySQL, MS-SQL Server (from Microsoft), DB2, Oracle, PostgreSQL, and other. In NoSQL, MongoDB is widely used right now. The other example of NoSQL are CouchDB, Redis, and other[5].

B. MongoDB

Mongodb is NoSQL database or we can call it as non-relational database. Before the NoSQL movement, database was saved in a relational or tabular form with rows and columns, which are related each other. Now, with NoSQL you can save the data in a database without thinking of making a single table. Because NoSQL is document-oriented, NoSQL database is schema-free. In MongoDB, there are collections. Collection is the name of organized in group of documents itself. The data in the collections is not related to each other. Although it's a schema-free database, but the document is based on JSON binary. The extension of the MongoDB's dump file is BSON. BSON stand for Binary and SON comes from JSON which is JavaScript Object Notation. In MongoDB, it is easy to upscale horizontally because there is auto-sharding process. So, in MongoDB there are 3 main terminologies; collection, document, and field.

In MongoDB, instead of using table, it is using collection. It seems like we can make an analogy that collection is a table in SQL or relational database, which is schema free. Inside the collection, there are one or more documents. In SQL or relational database management

system (RDBMS), a table can contain multiple columns and rows. In MongoDB, documents are replacing the rows. So the terminology or concept of MongoDB versus SQL are; collection is like the table, document is like the row, and field is like a column.

III. THE LEARNING CURVE

A. Installation

The installation step is clearly shown in MongoDB master manual. The MongoDB manual can be accessed from : https://docs.mongodb.com/manual/. As we can see in the manual, I do the exact steps of the installation MongoDB in Ubuntu[6] :

1. Import the public key used by the package management system
   ```
   sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A14518585931BC711F9BA15703C6
   ```
2. Create a list file for
   ```
   echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu xenial/
   ```
3. Reload local package database
   ```
   sudo apt-get update
   ```
4. Install the MongoDB packages
   ```
   sudo apt-get install -y mongodb-org
   ```

After finishing the installation, to run the MongoDB we need write `sudo service mongod start` and the MongoDB will run in the background system. To stop it, we can use command `sudo service mongod stop`. The next step to start using the MongoDB shell is to write command `mongo` after you start the MongoDB server before it.

B. Learning of the Syntax

In this section, will write the syntax or command that I already learn.

1. Create database : `> use DATABASE_NAME`
2. To show your current state, what database is in used : `> db`
3. To show list of database in local : `> show dbs`
   But, the list of database that will be shown is only the database which already has minimum one collection.
4. To drop a database, we need to use the database that we want to drop. After using the database, write the command : `> db.dropDatabase()`
5. There are two ways to create collection. Firstly you can create collection manually with command :
   ```
   > db.createCollection(name, option)
   ```
   Example :
   ```
   > db.createCollection("customer")
   > db.createCollection("customer", { autoIndexId : true})
   > db.createCollection("customer", { capped : true, size : 6142800, max : 1000})
   ```
   The option is an optional. This field required if you want to specify your collection or documents. There are 4 types of option :
   a. autoIndexId (value is a boolean : true/false) is needed if you want to create index in _id field when `autoIndexID : true`. The default value of this option is false.
   b. The capped (value is a boolean : true/false) is needed if you want to enable the capped database. Capped database is a database that replace the oldest entry if the whole of entries already reach their maximum capacity. If the capped's value is set as true, then, we need to specify the size.
   c. The value of the size in the option is an integer. It will specify the size of documents in bytes stored in a collection.
   d. The last option is max. Max's value is an integer. This option defines the number of maximum documents that are allowed in the collection.

   The second one, the collection automatically created if we insert one or more documents into a collection that hasn't been created before. The command will be like this
   ```
   > db.NameCollection.insert(Documents)
   ```
   As an example, the list of collection that already created are about customer and employee. After that, we run a command like this :
   ```
   > db.inventory.insert({name : barang, size : [
     p : 20
     l : 10
     uom : cm ]})
   ```
   then the collection inventory will be created automatically.
6. To show the list of collection :
   ```
   > show collections
   ```
7. To delete the collection :
   ```
   > db.NameCollection.drop()
   ```
8. To insert document into a collection, there are three ways :
   a. `> db.NameCollection.insertOne(document)` this command is used to insert only one document
   b. `> db.NameCollection.insertMany(document)` this command is used to insert many documents. If use insertMany, then the multiple documents must write down in array. The symbol of array is in a square brackets ([ ]).
   c. `> bd.NameCollection.insert(document)` this command is used to insert one or more documents

   When inserting document, the _id field can be automatically created as an ObjectId. This is the normal behaviour of inserting document in MongoDB. The ObjectId 12 bytes hexadecimal as an unique number that's consist of 4 bytes

timestamp, 3 bytes machine id, 2 byte process id, and 3 bytes incrementer.
Another way to insert document is by using command save(). The command save without specific _id field will have the same behaviour with insert. But if the _id field is specified, then the save() command will replace the whole data of document which has _id that has already defined.

9. To see all documents in a collections, we can use command :
   `> db.NameCollection.find()`
   But, sometimes the appearances are hard to understand. So we can add one command : prett() to make the appearances more tidy.
   `> db.NameCollection.find().pretty()`
   Furthermore, there's command findOne(), findOne() will only return first document in natural order. It usually mean the document with the smallest _id field.
   Both of find() and findOne() can be write with specific condition using query operator.
   In find() operator, if there's a nested query or a nested array, the sequential of parameters are matter and depending on it. But, if the parameters come with elemMatch, the sequence of parameters doesn't matter anymore.
10. There are two ways to update the document, we can use command :
    `> db.NameCollection.updateOne(SelectionCriteria, UpdateData)`
    or we can use
    `> db.NameCollection.updateMany(SelectionCriteria, UpdateData)`

    The difference of those mentioned ways are `updateOne()` will update only the first document that match with the criteria while `updateMany()` will update the whole document that match with the criteria. There are several operators to update such as
    - `$inc` : to increment the data in certain fields. If the value of inc is positive, the behaviour will be the same with sum operator. If the value of inc is negative, the behaviour will be the same with subtraction operator.
    - `$set` : is used to update field or insert new field in the documents
    - `$unset` : is used to remove field from the documents
    - `$addToSet` : is used to add member of array or add an array into the object or field. If we want to add multiple new member to array, we can use `$each`. So the syntax will become :

    `$addToSet : { NameField : $each {["mem1","mem2","mem3"]}}`
    - `$currentDate` : is a field that can be used to represent date or timestamp. The default type of `$currentDate` is date, this can be reach with boolean's value : true.

      example syntax :
      `$currentDate : {NameField : true}`
      Then, the NameField will updated to current date.
11. To delete the document we can use command :
    `> db.CollectionName.remove(DeletionCriteria)` this command will only remove first documment appear and match with the criteria.
    `> db.CollectionName.remove()` this command will remove whole of documents in that collection.

IV. THE COMPARISON SQL WITH NOSQL

This is SQL case that was used last semester. The schema of database is shown in Fig. 1 below.

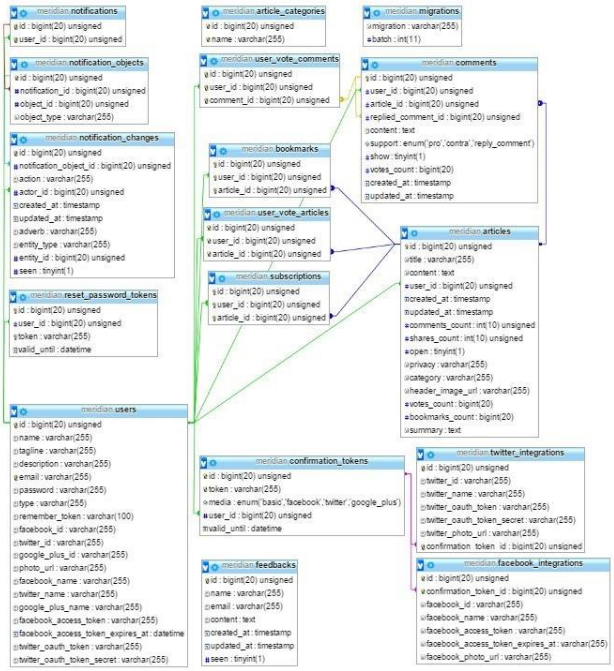

Figure 1 Schema Databes SQL

This schema is unnecessary if we used NoSQL. For the example in articles. In relational database, it needs 6 tables to store attributes related to the article, comment section, the categories of the article, bookmark, subscription, and user vote articles. In MongoDB, these tables can transform become only one collection. The following is the step that I create:

1. Create database

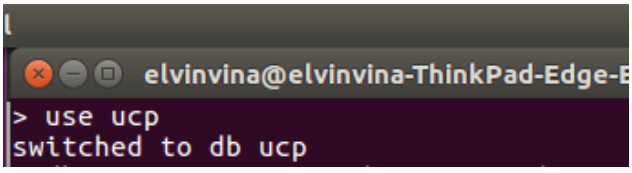

Figure 2 Create Database or Use Database that Already Created

2. Create collection articles

```
> db.createCollection("articles");
{ "ok" : 1 }
```

Figure 3 Create New Collection

3. Insert multiple documents

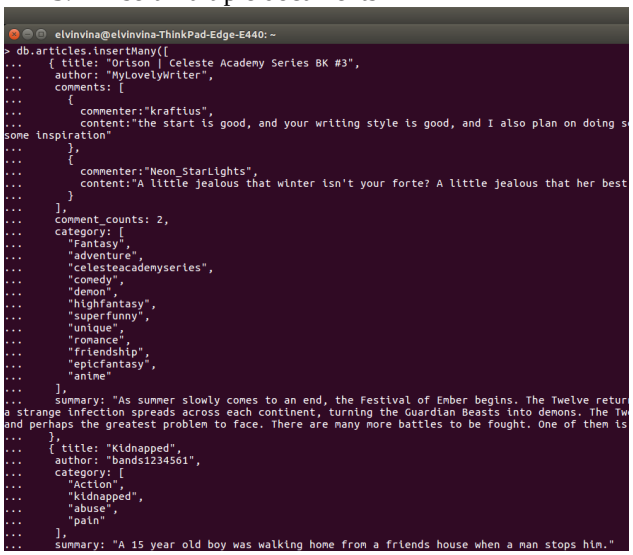

Figure 4 Insert Multiple Documents

The query that use when insert database is:

```
db.articles.insertMany([
  { title: "Orison | Celeste Academy Series BK #3",
    author: "MyLovelyWriter",
    comments: [
      {
        commenter:"kraftius",
        content:"the start is good, and your writing style is good, and I also plan on doing something like this in my book, so I will use some inspiration"
      },
      {
        commenter:"Neon_StarLights",
        content:"A little jealous that winter isn't your forte? A little jealous that her best memories aren't in summer?"
      }
    ],
    comment_counts: 2,
    category: [
      "Fantasy",
      "adventure",
      "celesteacademyseries",
      "comedy",
      "demon",
      "highfantasy",
      "superfunny",
      "unique",
      "romance",
      "friendship",
      "epicfantasy",
      "anime"
    ],
    summary: "As summer slowly comes to an end, the Festival of Ember begins. The Twelve returns to Celeste Academy with more questions as a strange infection spreads across each continent, turning the Guardian Beasts into demons. The Twelve are left with a strange riddle to solve and perhaps the greatest problem to face. There are many more battles to be fought. One of them is amongst one another"
  },
  { title: "Kidnapped",
    author: "bands1234561",
    category: [
      "Action",
      "kidnapped",
      "abuse",
      "pain"
    ],
    summary: "A 15 year old boy was walking home from a friends house when a man stops him."
  }
])
```

As you can see, the flexibility of MongoDB is shown there. One document with another document can have different field. Besides that, even though there is a same field, the length size of the array can be vary.

In this step, shows that the comment section can be written in a same collection with articles. Furthermore, we can use update the document until all of the tables from RDBMS accommodated in the new collection.

4. Update the documents to make all of table complete, so that the query is

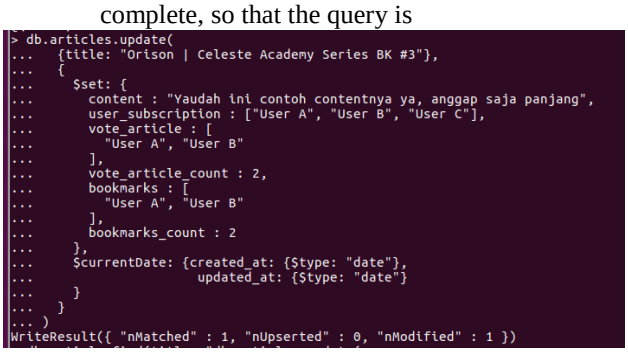

Figure 5 Update Documents to Complete the Field

The query that use when update the database is:

```
db.articles.update(
 {title: "Orison | Celeste Academy Series BK #3"},
 {
  $set: {
   content : "Yaudah ini contoh contentnya ya,
anggap saja panjang",
   user_subscription : ["User A", "User B", "User
C"],
   vote_article : [
    "User A", "User B"
   ],
   vote_article_count : 2,
   bookmarks : [
    "User A", "User B"
   ],
   bookmarks_count : 2
  },
  $currentDate: {created_at: {$type: "date"},
            updated_at: {$type: "date"}
  }
 }
)
```

In this query, the update was only use data dummy. The point is to show 6 tables in RDBMS can become 1 collection in MongoBD.

5. Update the document again as what needed

```
> db.articles.update(
...   {title: "Orison | Celeste Academy Series BK #3"},
...   {
...     $addToSet: {
...     bookmarks : "User C"
...     },
...     $inc: {bookmarks_count: 1}
...   }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Figure 6 The Queary That Use in Update Document as the Needed Field

The query that use is :

```
db.articles.update(
 {title: "Orison | Celeste Academy Series BK
#3"},
 {
  $addToSet: {
  bookmarks : "User C"
  },
  $inc: {bookmarks_count: 1}
 }
)
```

In this query, just simulated if the user that bookmark an article. This can be use as well to comment and voting too.

V. CONCLUSION

The goal of learning new tool is achieved, although whole of what I already learn cannot be represented this paper. Nevertheless, the main point of simplify and flexibility of the database structure especially in MongoDB is achieved and already implemented as the demo in above.

V. ACKNOWLEDGMENT

I would like to express my thankful to God who always protect me and give me ton of blessing so that I can finish this paper. And also, I would like to express my sincere gratitude to my teachers, Mr. Dr. Ir. Rinaldi Munir, M.T., Mrs. Dr. Eng. Ayu Purwarianti, S.T., M.T., and Mrs. Dr. Dessi Puji Lestari, S.T. M.Eng., who already though me the materials of IF3280 - Socio-Informatics & Professionalism and give the opportunity to write down this paper. To all my friends who give me the idea, help me write a good topics, support and give me the fate that I can finish this paper, and also proofread the paper, thank you so much.

REFERENCES

[1] Panwar, A. (2011). *Type of Database Management System*. Retrieved May 1, 2017, from http://www.c-sharpcorner.com/UploadFile/65fc13/types-of-database-management-systems/

[2] Eliot and the core MongoDB team. (2010). *State of MongoDB March, 2010* (para. 2). Retrieved May 1, 2017, from https://www.mongodb.com/blog/post/state-of-mongodb-march-2010

[3] NoSQL. Retrieved May 4, 2017, from http://nosql-database.org/

[4] Silberschatz, A., Korth, H.F., & Sudarsham, S. (2011). *Database system concepts, sixth edition*, p. 1. New York: McGraw-Hill.

[5] Issca, L. P. (2014). *SQL vs NoSQL Database Differences Explain with few Example DB.* Retrieved May 1, 2017, from http://www.thegeekstuff.com/2014/01/sql-vs-nosql-db/?utm_source=tuicool

[6] The MongoDB 3.4 Manual (pp. 85-88). Retrieved April 23, 2017, from https://docs.mongodb.com/manual/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Mei 2017

Elvina R. K. Situmorang (13514045)