*International Journal of Internet, Broadcasting and Communication Vol.10 No.3 51-58 (2018)*
*http://dx.doi.org/10.7236/IJIBC.2018.10.3.51*

**IJIBC 18-3-6**

# Modeling and Implementation of Public Open Data in NoSQL Database

Meekyung Min

*Dept. of Computer Science, Seokyeong University, Seoul, Korea*
*mkmin@skuniv.ac.kr*

***Abstract***

*In order to utilize various data provided by Korea public open data portal, data should be systematically managed using a database. Since the range of open data is enormous, and the amount of data continues to increase, it is preferable to use a database capable of processing big data in order to analyze and utilize the data. This paper proposes data modeling and implementation method suitable for public data. The target data is subway related data provided by the public open data portal. Schema of the public data related to Seoul metro stations are analyzed and problems of the schema are presented. To solve these problems, this paper proposes a method to normalize and structure the subway data and model it in NoSQL database. In addition, the implementation result is shown by using MongDB which is a document-based database capable of processing big data.*

***Keywords:*** *Public Open Data, Big Data, Seoul Metro, NoSQL, MongoDB*

## 1. Introduction

Korea Public Open Data Portal [10] is an integrated site that provides open data, which are generated or acquired by public agencies, in one place. Public data portals are integrated gateways that provide public open data that a public entity creates, acquires and manages in. In recent years, individuals and institutions have been increasingly using public data to analyze and visualize it [3, 4]. The portal provides various data types such as files, open APIs, data visualizations to allow you use open data easily. Most of the file data is in the form of csv, xls, json and xml. The open API takes the parameters from the user, extracts the data that the user wants from the files, and provides it as an xml file. Some file data may be stored in the database. In this case, the data is extracted from the database.

However, current public open data has the following problems. First, since there is duplication of information between files, it is difficult to maintain consistency of data. In addition, it is difficult to efficiently manage the data separated into many files. This problem can be solved if the DBMS stores and manages the file data in the database. Second, the schema of public data is not normalized, so new data modeling is needed. Third, public data will become increasingly cumulative and contain large amounts of data in the future. Especially, since the amount of dynamically generated data increases exponentially, a

Manuscript Received: June. 5, 2018 / Revised: June. 11, 2018 / Accepted: June. 17, 2018
Corresponding Author: mkmin@skuniv.ac.kr
Tel: +82-2-940-7750, Fax: +82-2-919-0345
Dept. Of Computer Science, Seokyeong University, Seoul, Korea

method of processing big data is needed. To satisfy all of these needs, we can consider introducing a database capable of big data processing.

In this paper, we model public data with document-based NoSQL data model and implement it in MongoDB. The target public data is subway-related data provided by Seoul Transportation Corporation. The subway data includes static data about the subway or the station itself, such as the related facilities and locations. It also contains a lot of dynamic data about the subway operations or the passenger boarding information. Therefore, big data processing is necessary for the subway data and a study on public big data management using NoSQL database will be meaningful.

This paper is composed as follows. Chapter 2 describes the related works for NoSQL database and MongoDB. Chapter 3 analyzes the schema of subway-related public data and identifies some problems. In addition, the normalization and structuring methods to solve these problems are mentioned. Chapter 4 shows the result of modeling the subway data as document-based NoSQL DB and storing it in MongoDB. Chapter 5 concludes this study.

## 2. Related Works

### 2.1 NoSQL

A new database called NoSQL is steadily evolving to address the weaknesses of relational databases. NoSQL databases are widely spread and have the following characteristics: NoSQL does not use SQL and works without a schema. It is usually an open source project, mostly intended to be run on a cluster. It is usually based on the needs of the web environment in the early 21st century [1].

The biggest advantage of NoSQL is the application development productivity. In a relational database, many application development efforts are spent mapping in-memory data structures to relational databases. The NoSQL database simplifies the interaction between applications and databases by reducing these object-relational mismatches. Next, NoSQL is appropriate when many organizations collect and process more data quickly. Relational databases are expensive to perform these tasks. Many NoSQL databases are designed to run in clusters and are therefore better suited for big data [2, 8].

NoSQL databases are classified into four categories according to the data model: A key-value database, a document database, a column family store, and a graph database. The key-value database is a simple hash table used to access the database only through the primary key. Popular key-value databases include Redis, Memcached, and BerkeleyDB. The document database includes couchDB and MongoDB. Documents such as XML, JSON, and BSON are key concepts in document-based databases. The document database can be thought of as a key-value store for checking values. Column family stores include Cassandra, HBase, and HyperTable. The store stores data corresponding to the key value. The values are divided into multiple column families, where each column family is a map of the data. In the graph database, entities are called nodes and have attributes. Several nodes are connected in various relationships and represent data in relation to the nodes. Such databases include Hypergraph DB and Neo 4J [6, 7].

### 2.2 MongoDB

MongoDB [9] is one of the most representative NoSQL databases. MongoDB is a free and open-source cross-platform document-oriented database. MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc., and is published under a combination of the GNU Affero General Public License and the Apache License [5].

MongoDB is a document database. A record in MongoDB is a document, which is a data structure

composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents. The advantages of using documents are: Documents (i.e. objects) correspond to native data types in many programming languages. Embedded documents and arrays reduce need for expensive joins. Dynamic schema supports fluent polymorphism, high performance, high availability, rich query language, and automatic scaling [6].

MongoDB is released as two editions: Community and Enterprise. The user interface for MongoDB is the mongo shell. Mongo shell is used to query and update data as well as perform administrative operations. MongoDB Compass is the GUI for MongoDB.

## 3. Analysis of Public Open Data

### 3.1 Schema

There are many data related to Seoul Metro, and researches and applications on it are increasing. In addition, since the number of subway passengers in Seoul has steadily increased by more than one million over the last 10 years, it is meaningful to analyze subway data from various angles.

This chapter describes the analysis results of the Seoul metropolitan subway data in the Korea open data portal [10]. The original information is provided by Seoul Transportation Corporation and it is also provided by Seoul Metropolitan Government. This paper uses subway information provided by Seoul Transportation Corporation. The data are in the form of several types of files such as csv, xls, json, and xml. The files in different format can be converted to each other, so they can be viewed as the same file. Table 1 shows the analyzed schema of the Seoul metropolitan subway data. The data can be classified into three categories. There are many files, and the schema of the file is the fields that make up the file. Actual data consists of field and value pairs.

### 3.2 Normalization

When analyzing the schema of each file in Table1, there are problems such as inconsistency or duplication, and it should be normalized. The problems are as follows.

First, since the files are often unnecessarily separated, the associated files must be merged. If the current file data is classified into groups, it can be classified into *subway passenger transportation data*, *subway train operation data*, and *subway station data*. Since most of the data in all groups is information about one station, they can be integrated into one. Second, information is redundant. For example, the *station number*, *station name*, and *line number* are repeated for each file. The representative field that identifies one station is sufficient for the *station number*, so redundancy should be removed. If you convert the file data into a relational database, this representative field will be the primary key and the foreign key. Third, statistical data and stored data exist at the same time. For example, the *ranking of the number of daily passengers by station*, the *number of passengers per month by year*, the *number of transit passengers by year*, and the *number of passengers and transportation by year* are statistical data that can be obtained from the stored data, which is the *number of passengers of boarding / alighting by time of day according to station*. Statistical data can be derived from stored data, so statistical data need not be stored. However, whether or not to keep the stored data and the statistical data redundantly should be determined in the data modeling. In some cases, statistical data and stored data coexist in a single file. An example is the *station facilities* file. In this case, statistical data should be removed and normalized. Fourth, field names and values are inconsistent. In the case of the station name of "Seoul Station", it may be a string like "Seoul Station", or there may be a *line number* in parentheses like "Seoul Station (1)". In some cases, there may be a *station number* in parentheses,

such as "Seoul Station (150)".

Normalization is necessary to solve the above problems. Normalization can be performed while maintaining data in the form of a file, or can be solved by converting file data into a database. The first and second problem is the duplication of information between files. This is part of the problem when using a non-DBMS file system. Assuming you are using a relational database, all of these problems should be eliminated. However, when using file data, it should be designed so as to reduce duplication if possible. The third problem on statistical data is a design issue. Both in the case of using file data and in the case of using database, whether to store statistical data or not is to be determined according to the data design direction. For the last problem, data types in the data field, must be normalized so that the general users can easily use the file data and the application programmers can conveniently develop the application programs using the file data. The data type must be normalized as well as when using the database.

### 3.3 Hierarchical Structuring

In addition to normalization, there is a layering problem in public data. In other words, it is a matter of how the structure of the public data is to be made.

**Table 1. Schema of Seoul Metropolitan Subway Data**

| Category | File name | Fields in the file | Format |
|---|---|---|---|
| Subway passenger transportation data | Number of passengers of boarding/ alighting by time of day according to station | Station name, Date, boarding/alighting, 05-06, 06-07, 07-08, ..., 00-01 | csv, xml |
| | Ranking of the number of daily passengers by station | Ranking, Station name, Daily average | csv, xml |
| | Congestion level by station | Day(weekday/Sat/Sun), Line number, Direction, Station name, Station number, 05, 06, 07, …, 24 | csv, xml |
| | Number of passengers per month | Station number, Station name, Jan, … , Dec | xls |
| | Number of passengers by year | Year, Line1, Line2, Line3, Line4 | csv, xml |
| | Number of passengers and transportation by year | Year, Number/Income, Line1, Line2, Line3, Line4 | csv, xml |
| | Number of free passengers by station | Year, Total, Senior, Handicapped, Veterans, Ratio, Converted fare, | csv, xml |
| | Number of complimentary tickets by station | Station name, 2006, 2007, …, 2015 | csv, xml |
| Subway train operation data | Subway train timetable(separate files by station) | Train number, Start station, End station, Station name, Arrival time, Departure time | xls |
| | Subway last train timetable | Station code, Station id, Station name, End station name, Start time, Day, Direction | csv |
| | Seoul Metro train service information | Title(Section, Operating distance, No of stations, Time, No of trains, No of services, Speed, Train mileage, Interval, Operation number), Unit, Total, Line 1, Line2, …, Line8 | xls |

**Table 1. Schema of Seoul Metropolitan Subway Data (continued)**

| | | | |
|---|---|---|---|
| Subway station data | Facilities | Station name, Nursing, Bike slope, Unmanned delivery box, ATM, Snack vending machine, Photo box, Petition documents machine/ office | csv, xml |
| | Toilets | Station name, No of toilets, No of toilets inside the station, No of toilets outside the station, No of toilet guide plates | csv, xml |
| | Station facilities | Station name, No of card charging machine, No of deposit refund machine, No of card vending machine, No of elevators, No of escalators, No of gates, No of speed gates, etc. | csv, xml |
| | Bus stops around station | Station code, Station number, Station number, Line number, Bus stop name, Bus stop id, X-coordinate, Y-coordinate | csv, xml |
| | Station address | Station name, Line number, Gu, Dong, Full address, Phone number | csv, xml |
| | Station Multilingual name | Line number, Station name, Chinese, English, Japanese | |

Currently, the structure of public data is flat and not hierarchical. Some of the public open data are in xml and json format, but this does not reflect the features of the xml or json document structure. That is, no fields are nested and do not form a nested hierarchy, and they are composed of one layered hierarchy. It basically comes from the original data in the form of a csv or xls file. Therefore, the files provided by the public data portal can be regarded as the same file although they are in different formats. The same is true for the open API's. However, if we look at the schema of public open data, nested structure is more natural than flat structure of one level. In the case of data separated into multiple files, it is much easier to access the data if each file consists of a lower layer of another file as a document.

Based on the results of this analysis, this paper suggests a method of constructing the subway-related public open data by normalizing it and layering it in json form.

## 4. Modeling and Implementation

### 4.1 Modeling in NoSQL

In this paper, subway information data is modeled using document-based NoSQL data model. In a NoSQL database, the data consists of documents very naturally. Within a single document data structure, related sub-documents and arrays are embedded. Thus, the database matches the programming language naturally. Figure 1 shows an example of modeling subway public data as a document named *Station*. It shows the connected objects centered on the document object "Euljiro-1-ga". *Euljiro-1-ga* consists of another sub-document or an array of documents. The *Boarding information* and the *Alighting information* of one station are arrays, which are again composed of other documents and arrays. The same is true of *Train Timetable*. *Bus Stop* is an array of bus stops around the subway station. In the case of a *Bus Stop*, this is a separate document object, which the *Station* refers to as a field. Although omitted in Figure 1, there are other fields, such as facility, address, and line number, and etc.

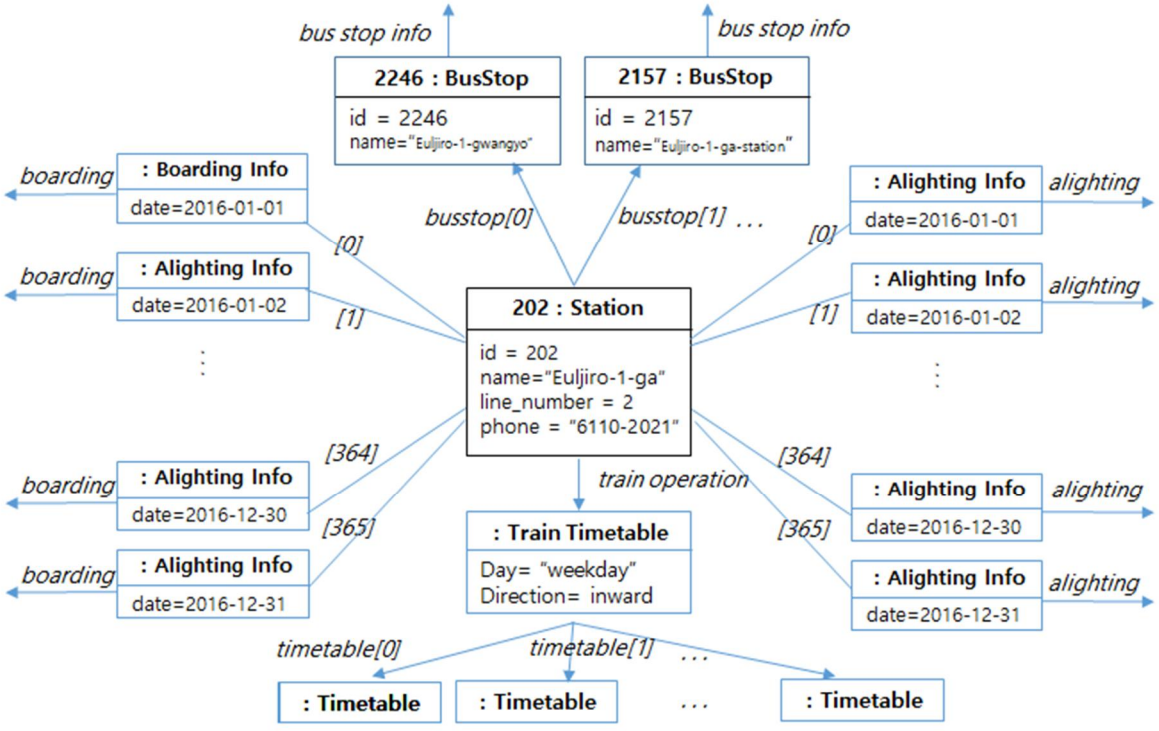


**Figure 1. Sample Modeling of Subway Data**

### 4.2 Implementation with MongoDB

It is desirable to use a database capable of big data processing in order to store public open data that increases dynamically. In the case of subway data, the amount of data is very large because the number of passenger data and train operation data continuously increase. In order to analyze and utilize accumulated big data, a database for big data should be used. In this paper, the subway data modeled in Figure 1 was stored using MongoDB, the most widely used NoSQL database. MongoDB is a document-based database that stores data in document form, so it is suitable for subway data structure. MongoDB used in the implementation is MongoDB 4.0 as open source.

The procedure for inserting data into the database is as follows. First, the csv file downloaded from the public data portal as shown in Table 1 was used as input data. Next, I wrote a Python program that converts this csv files to json files. The Python program parses the input csv file, defines a json document structure in the form of a data model as shown in Figure 1, and creates a json file accordingly. Next, this json file is input into the database using the mongo shell, which is a MongoDB command line interface.

Figure 2 shows the contents of a subway station document stored in the database. The GUI shown in the figure is MongoDB Compass. The DB name is *SeoulMetro*, and the collection name is *Station*. There are documents in the collection *Station*. One document is created per station and there are 254 stations, so 254 *Station* documents are created. Figure 2 shows two of these, *Euljiro-1-ga* and *Samsung*.

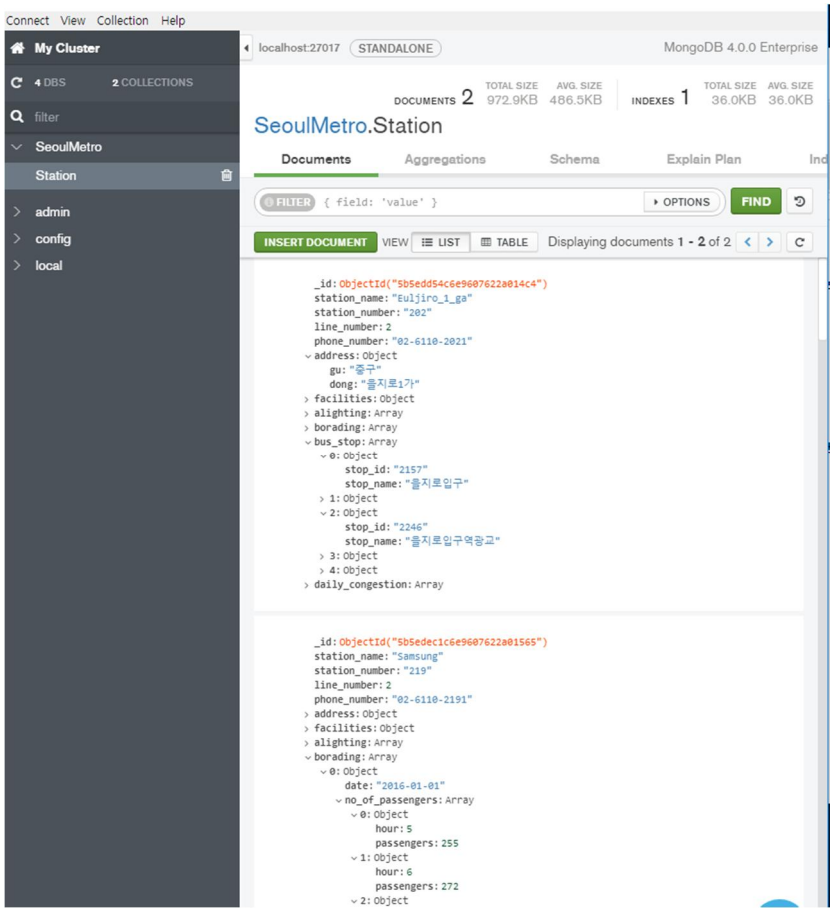


**Figure 2. MongoDB Implementation of Subway Database**

## 5. Conclusion

This paper presented a method to model public data provided by Korea open data portal in NoSQL and to implement it using document-based NoSQL database MongoDB. The target data is the public data related to the subway. As a result of analyzing the subway public data, the following problems were found. Files are unnecessarily separated and information is redundant. In addition, statistical data and stored data were duplicated, and the names and values of the fields were inconsistent. In the structural problem, since the data is structured in a flat structure, it is understood that it will be better mapped to the original data if it is structured hierarchically. To solve these problems, this paper proposed a method to normalize and structure the data, and also showed an example of metro public data in NoSQL format. Finally, the subway public data modeled by NoSQL is stored and implemented in MongoDB. It was also used to write Python programs.

The result of this paper can be used as a basic reference for managing the public big data accumulated in the future by using the database. It can also be used when analyzing public Big Data to implement an application system. It is a future task to further implement a system that generates the results requested by the user from the public data stored in the database. If the future tasks are completed, this study may be more useful.

## References

[1] P. Atzeni, F. Bugiotti, L. Cabibbo, and R. Torlone, "Data modeling in the NoSQL world," Computer Standards & Interfaces, Elsevier, 2016.
DOI: https://doi.org/10.1016/j.csi.2016.10.003.

[2] D. Chandra, "BASE Analysis of NoSQL database," Future Generation Computer Systems, Vol. 52, pp. 13-21, Nov. 2015.
DOI: https://doi.org/10.101/j.future.2015.05.003.

[3] J. Chang, "An Experimental Evaluation of Box Office Revenue Prediction through Social Bigdata Analysis and Machine Learning," The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 17, No. 3, pp. 167-173, June 2017.
DOI: https://doi.org/10.7236/JIIBC.2017.17.3.167.

[4] S. Kim, "A Study on Construction of Crime Prevention System Using Big Data in Korea," The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 17, No. 5, pp. 217-221, Oct. 2017.
DOI: https://doi.org/10.7236/JIIBC.2017.17.5.217.

[5] M. Min, "Experiments of Search Query Performance for SQL-Based Open Source Databases," International Journal of Internet, Broadcasting and Communication(IJIBC), Vol. 10, No. 2, pp. 31-38, May 2018.
DOI: https://dx.doi.org//10.7236/IJIBC.2018.10.2.6.

[6] P. Sadalage and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Addison-Wesley Professional, 2012.

[7] Jing Han, Haihong E, Guan Le, and Jian Du, "Survey on NoSQL Database," in Proc. 6th International Conference on Pervasive Computing and Applications, pp. 363-355, Oct.26-28, 2011.

[8] A. Schram and K. Anderson, "MySQL to NoSQL: Data Modeling Challenges in Supporting Scalability," in Proc. 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity, pp. 191-202, Oct.19-26, 2012.
DOI: https://doi.org/10.1145/2384716.2384773.

[9] MongDB. *https://www.mongodb.com.*

[10] Korea Open Data Portal. *https://data.go.kr.*