4th International Conference on Applied Research in
ENGINEERING, SCIENCE AND TECHNOLOGY
20-22 August, 2021 Dublin, Ireland

A Tool for Generating Test Data for Document-Oriented Database

Chulaluk Bumrungvet[1], Taratip Suwannasart[2]

[1,2] *Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand*

Abstract.

Document-oriented databases are widely used in numerous applications since they can store all types of data and do not require predefinition of the data type or structure. For testers, generating test data with existing tools will result in relational data with fixed structure. This may not be compatible with the nature of document-oriented databases, which can store various type of data structures. This paper presents a test data generation tool for document-oriented database called NSTG, a tool that allows testers to generate test data in different forms and compatible with document-oriented database. The tool can connect to a test database to analyze the existing data and extract the database schema. Testers can modify the database schema to generate test data that is consistent with the real data. The experimental results of the tool demonstrated that the tool can generate test data that is compatible with document-oriented database and consistent with the actual use.

Keywords: Database Testing; MongoDB; NoSQL; Software Testing; Test Data Generation

1. Introduction

In software testing, software testers design test cases that are executed to identify software defects. Preparing test cases sometimes requires software testers to create test data. To ensure the integrity of the software behavior, the test data is naturally prepared to cope with the designed test cases. If the test data is well designed for certain database characteristics, it can significantly affect the efficiency of software testing. Therefore, it is important to design test data properly.

NoSQL databases are gaining a positive reputation in present due to the flexibility in storing data and the ability to handle complex data structures. According to the ranking of database popularity on the DB-Engines website [1]. MongoDB was ranked at the top of the list as the most popular NoSQL database. There are several database categories in the field of NoSQL databases. MongoDB was categorized as a document-oriented database [2]. It stores data as an entity called document, and each document does not have to contain the exact same fields or data types. This flexibility gives MongoDB the ease to handle any form of data without considering the structure of the data being stored.

Synner [3] and Mockaroo [4] are examples of tools for generating test data for NoSQL databases. They can export the result to a file. Software testers can use this file to insert the generated test data into the target database. However, the schema of test data generated by these tools are identical. When using these tools to generate test data for database, software testers must specify the data structure into the tools to match with the target database manually since the tools are not able to connect to the database and analyze the schemas for them. The result of studying these tools shows that the usage and outputs are not compatible with flexible databases such as NoSQL databases.

In related studies, Kritsana Piriyakitpaiboon and Taratip Suwannasart proposed a tool to generate test data based on the constraints from the target database and user input [5]. The test data was generated according to the target database schema and logical dependency between data fields of each table. However, the proposed tool has a major limitation. It does not fully support NoSQL databases and only supports relational databases; A database type that has a fixed amount of attribute, type of each attribute, and schema.

This paper presents a test data generation tool for document-oriented database called NSTG, that aims to aid software testers in generating test data with ease. The tool can either connect to MongoDB databases or receive user input, then analyzes fields, field types, and field constraints. In the end, the tool will flexibly generate test data from this analyzed information. The tool can also export test data results to the target MongoDB database directly or to JSON files.

The remainder of the paper is organized as follows. Section 2 provides background. Section 3 describes the process of the tool. Empirical studies are discussed in section 4. Section 5 provides the conclusion and the last section deals with future work.

2. Background

2.1 Test Data

Test data [6] is existing data or creating data based on the designed test cases. There are various types of test data depending on the type of test cases. For example, test data in the form of files, test data stored in the database, etc. Testers generally prepare these test data before start testing the software under test (SUT). They can use test data to ensure that the integrity of SUT meets expectations, or to measure how SUT performs when facing outlier data of the software.

2.2 MongoDB

MongoDB database [7] is a document-oriented database system for storing a large amount of unstructured data. The MongoDB database stores data in the form of collections and documents, where a collection contains various documents, like how a table contains data in a relational database system. A document is the smallest unit of data storage and is formatted as a JSON object. In addition, the document consists of a field and a field value.

2.3 JSON

JSON [8] is a human-readable format for storing and exchanging data. It is popularly used for transmitting data between the client-server communication. JSON typically has 2 parts which are the keys and the values. The constraint of the keys is that they must be written in between quotation marks (“ ”). The values, however, have various data types such as String,

Integer, Boolean, or Object. They can be expressed in multiple ways depending on the data type. Finally, between the keys and the values must have a colon (:) to separate them.

2.4 JSON schema validation

JSON schema validation [9] is one of MongoDB features that validates the data structure integrity during data insertion or amendment. It ensures that every document within the same collection complies with the validation rules set by the user. MongoDB database supports validation rules configuration via Operator or JSON Schema. For example, we could indicate the properties of the field 'name' to have the minimum length as 1 and the maximum length of this field as 30. A JSON Schema sample is shown in Figure 1.

Figure 1: A sample of validation rules configuration

```
$jsonSchema : {
    properties : {
        name : {
            minLength: 1,
            maxLength: 30
        }
    }
}
```

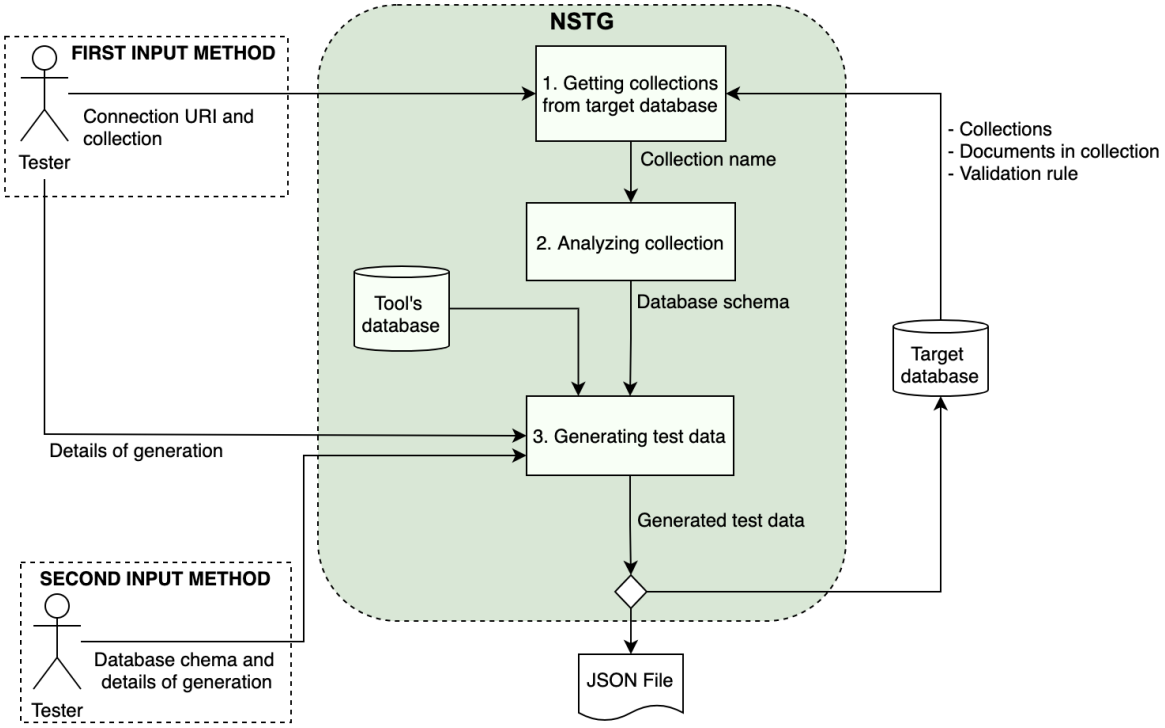
Source: Authors

3. NSTG

NSTG (Non-SQL test data generator) is a tool that was developed to generate test data for testing SUT that uses document-oriented database. NSTG generates test data by referring to the database schema and details of test data generation. For database schema, a tester can define it in 2 methods. First, the tester must enter a database connection URI and select a target collection. NSTG will then get the database schema by analyzing documents and validation rules in the target database. NSTG can analyze for database schema only when connecting to MongoDB directly. Alternatively, the tester must define the whole details of the database schema which are field name, field type, and field constraint.

In the test data generating step, the tool requires 6 properties for generating test data. These properties are field names, field types, field constraints, field generating patterns, value-generating methods, and output document amount. NSTG provides 6 field types which are string, double, boolean, integer, long, and decimal. NSTG provides 3 generating patterns which are the "all-document" pattern, "specific document" pattern, and "percentage" pattern. NSTG can generate field values from 3 methods which are the "randomly" method, "constant" method, and "general data" method. The process of NSTG is shown in Figure 2

Figure 2: The process of NSTG

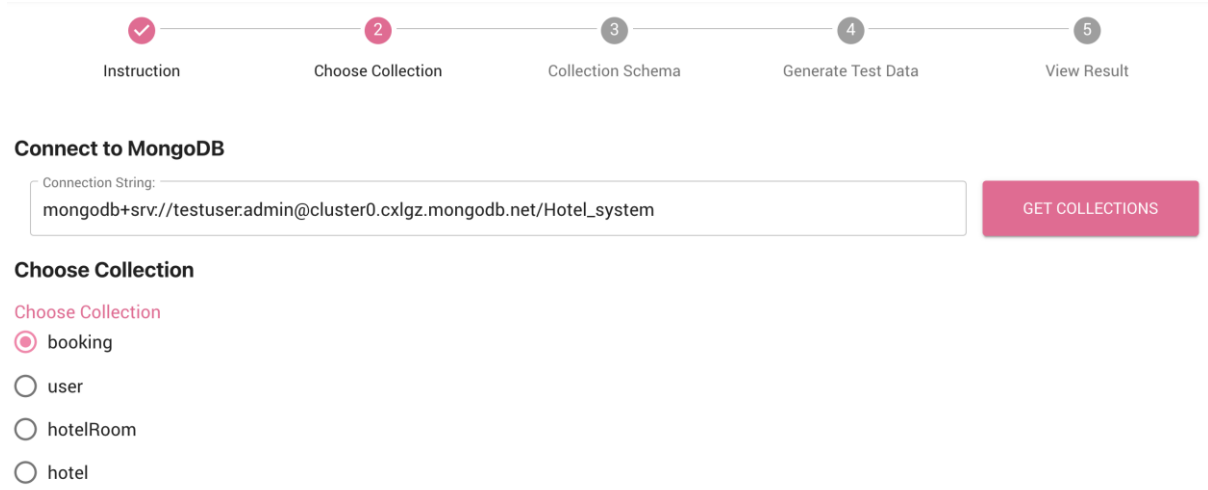


Source: Authors

A. Getting collections from target database

The first step is to connect to the target database using a database connection URI from the tester. After connected successfully, NSTG will display a collection list within this database. Thus, the tester is responsible for selecting the collection that NSTG will be analyzed as shown in Figure 3.

Figure 3: A sample of connected database and collection list



Source: Authors

B. Analyzing collection

In this step, NSTG will analyze every single document within the selected collection to build its database schema, which can be broken down into two steps. First, NSTG will collect field names and field types from all documents in the collection. In the case of multiple data types in a single field, NSTG will also collect all the data types as a field type. Finally, NSTG will analyze the validation rules to collect the constraints of each field.

C. Generating test data

Before proceeding to the generating test data process, the tester needs to specify the details that are required by the process. The details required by the process are generating pattern, generating method for field value, and a number of generated documents.

First step, the tester must specify the generating pattern for each field. NSTG provides 3 generating patterns which are "all-document" pattern, "specific document" pattern, and "percentage" pattern.

- The all-document pattern means that every document will contain this field.
- The specific document pattern means that the tester can select specific documents by entering a document number or range. The tool then generates the field only for those documents. For example, in Figure 4, the tester specifies the ‘totalPrice’ field has the pattern as a range of documents from 1-5, meaning that NSTG will create a ‘totalPrice’ field for the first 5 documents from number 1 to number 5 only.
- The percentage pattern means that only a certain number of documents proportional to the specified percentage will contain on this field. For example, the tester wants to create 10 test data documents, and the tester selects the percentage pattern for ‘hotelName’ field by entering value as 50 percent. NSTG will randomly select at 50 percent from 10 documents which mean 5 documents are selected, then create them with ‘hotelName’ field included.

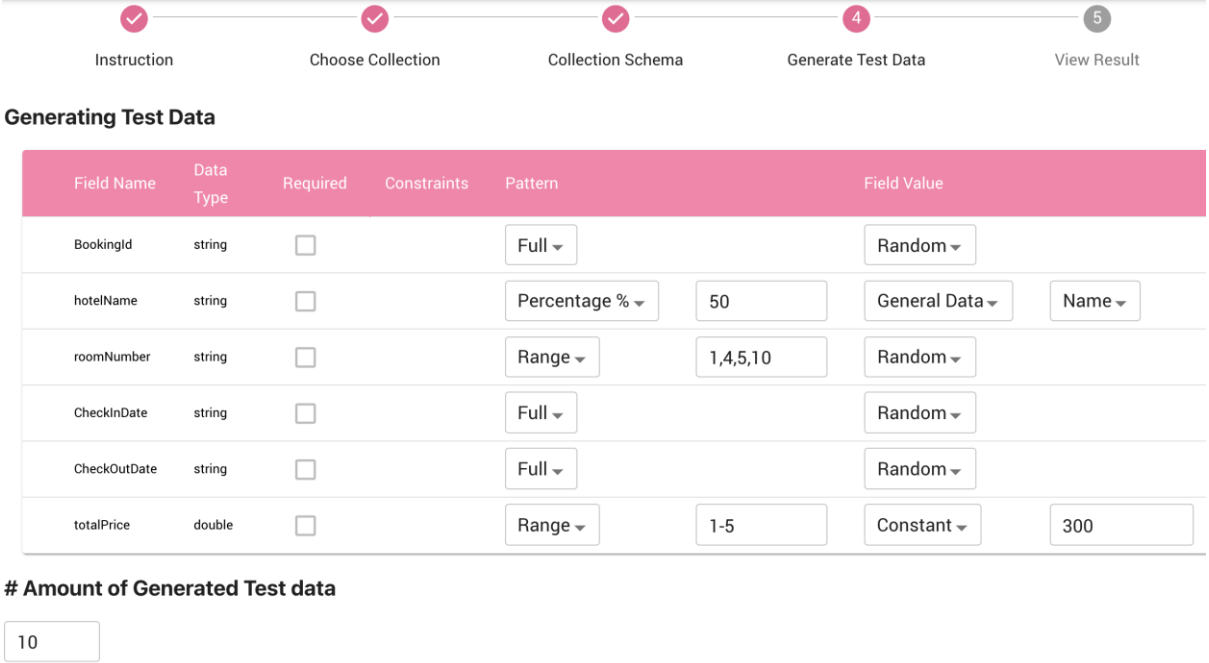
Second step, the tester needs to define the method to generate field value of each field. NSTG

can generate field values from 3 methods which are "randomly" method, "constant" method, and "general data" method.

- The random method means that each field value will be randomly generated based on field type and field constraints.
- The constant method means that the tester can specify a constant value to the whole field.
- The general data method means that NSTG will randomly pick from the given general dataset chosen by the tester. This general dataset is fully customizable by the tester and will be saved to NSTG's database.

Third step, the tester needs to define the number of documents to be generated. A Sample of details of generation is shown in Figure 4.

Figure 4: A Sample of amount, selected pattern, and selected field value method of each field.



Source: Authors

In the test data generation process, NSTG will generate test data according to the database schema, test data amount, and generation details of each field respectively. The test data generation starts by generating each test data document, one field at a time based on the given details. The process stops when the number of output documents meets the expected amount. After the test data generation process has been completed, the tool displays the generated test data results on the table as shown in Figure 5.

Figure 5: A sample of the generated test data results

Generated test data = 10 Documents

#	BookingId	hotelName	roomNumber	CheckInDate	CheckOutDate	totalPrice
1	CwI5h0YY	Nan	1Hgnu3G	lq1svYVz61	GB9	300
2	bs4T	Pumpkin	<no-field>	IV	gH	300
3	fgqqcS	<no-field>	<no-field>	vbo7JZL	hO9rRFdCXi	300
4	EfxUpkE	<no-field>	I3af1ZjSr	kL	ZxGlP7YEV	300
5	EQI	<no-field>	wXcX4McgIH	J86ef	hcug1	300
6	eFHxoDd	Somchai	<no-field>	PjLYy33tD	cKI	<no-field>
7	UlU9bG1	Pumpkin	<no-field>	CaPn2Y2	PtQ	<no-field>
8	NYFN	Somsri	<no-field>	Bvh	wF6iTu2	<no-field>
9	xotAB1	<no-field>	<no-field>	oFR2pWv	19M12X2	<no-field>

SAVE JSON FILE IMPORT TO DB

Source: Authors

Tester can choose between 2 methods to save the generated test data output. Either download as a JSON file where the output data will be written in JSON format and popup a download dialog or insert into the target database where NSTG will automatically import the test data into the destination database directly.

Limitations

NSTG supports only MongoDB database, while there are numerous document-oriented databases on the market today such as CosmosDB, CouchDB, RavenDB, etc. Moreover, NSTG supports only 6 data types, while document-oriented databases can support more than 6 data types, such as arrays, objects, dates, etc. Thus, when NSTG analyzes any unsupported data type within the collection, it will automatically set 'object' as data type by default. NSTG cannot generate test data for multiple collections in one process. The tester can manually create the database schema or retrieve it by analyzing a database, but the analyzed schema is immutable. Finally, to import generated test data, NSTG would import the data to an existing collection only. Creating a new collection from the tool itself is not supported.

4. Empirical studies

The tool is applied to generate test data for three software systems: "Restaurant Management System", "Hotel Booking System", and "Employee Management System". The tool was connected to the target database to analyze their collections. The results indicated that the tool could analyze the collections to get database schema and generate test data based on constraints from the database schema of SUT and details of generation.

Another empirical study of NSTG is to measure the response time of analyzing collections in the target database. In reality, the number of documents and collections on the current database system is excessive. The process of analyzing the database schema requires NSTG to go through all documents and validation rules. Additionally, it is important to consider the response time when analyzing the collections. An experiment was set up and conducted on a computer notebook that has a 2.3GHz Intel Core i5 processor and 8 GB of RAM. The experiment to measure the response time from analyzing collection contains 3 factors, which are the number of fields in the document, whether validation rules are present or not, and the data type of each field. Table 1 shows the result of this experiment.

Table 1: The result of response time experiment

Test case	Number of documents in the collection	Number of fields	Has validation rules?	Data type of each field	Response time (Second(
1	10	5	No	All the fields used the same data type.	0.097
2	10	5	No	All the fields used different data type.	0.504
3	10	5	No	Some fields have mixed data types.	0.649
4	10	5	Yes	All the fields used the same data type.	0.111
5	10	5	Yes	All the fields used different data type.	0.474
6	10	5	Yes	Some fields have mixed data types.	0.656
7	10	10	No	All the fields used the same data type.	0.100
8	10	10	No	All the fields used different data type.	0.810
9	10	10	No	Some fields have mixed data types.	0.977
10	10	10	Yes	All the fields used the same data type.	0.095
11	10	10	Yes	All the fields used different data type.	0.853
12	10	10	Yes	Some fields have mixed data types.	1.006

Source: Authors

According to Table1, the results indicated that when all fields had the same type, the response time was significantly decreased. On the contrary, the response time was incredibly increased if each field is different from the other. Moreover, the collection with verification rules took slightly more response time compared to the collection without verification rules. However, the amount of field within the collection does not affect the response time of analyzing the collection. Hence, the response time of analyzing collections with NSTG is acceptable.

5. Conclusion

This study introduces NSTG, a tool to generate test data for a document-oriented database that currently supports MongoDB database only. NSTG can analyze all the data and validation rules within the selected collection to extract the database schema out of it. This database schema can be used to create new test data. Moreover, the tool also supports manually input database schema when connecting to the target database is not an option. To generate test data, the tester must provide the amount of data to be generated, the patterns of each field, and the field value method of each field. Once the generation was done, the tester can save the generated data to the JSON file or save it to the connected database directly.

6. Future work

There are several directions for future work. First, the researchers will increase the data types that NSTG supports. Second, the researchers will improve the ability to manage the connected target database, for instance, collection creation. Third, NSTG will support document-oriented databases other than MongoDB. Finally, NSTG will supports other types of file when exporting test data as a file to download, e.g. CSV file and XML file.

References

[1] Solid IT gmbh (June 2020). DB-Engines Ranking - Trend of MongoDB Popularity. [Online]. Available: https://db-engines.com/en/ranking_trend/sys tem/MongoDB

[2] Jose, B. & Abraham, S. (2017). Exploring the merits of nosql: A study based on mongodb. *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*. Thiruvananthapuram, India, pp 266-271.

[3] Mannino, M. & Abouzied, A. (2019). Is this Real?: Generating Synthetic Data that Looks Real. *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. New Orleans, LA, pp 549–561.

[4] Mockaroo (June 2020). *Realistic data generator*. [Online]. Available: https://www.mockaroo.com/

[5] Piriyakitpaiboon, K. & Suwannasart, T. (2004). *RealGen: A Test Data Generation Tool to Support Software Testing*. *Proceedings of the 2004 International Conference on Communication Technologies*. Bangkok, Thailand, pp. 203-210.

[6] Homes, B. (2012). *Fundamentals of Software Testing*, 1st ed. London, UK: Wiley-ISTE.

[7] Tiwari, S., (2011). *Professional NoSQL*, 1st ed. Indianapolis, MI: John Wiley & Sons.

[8] *Introducing JSON* (n.d.). Json. [Online]. Available: https://www.json.org/

[9] *MongoDB Documentation: Schema Validation* (n.d.). Schema Validation - MongoDB Manual. [Online]. Available: https://docs.mongodb.com/manual/ core/schema-validation/