Semi-Structured Data Model for Big Data (SS-DMBD)

Shady Hamouda[1,2] and Zurinahni Zainol[1]
[1]*Universiti Sains Malaysia, Penang, Malaysia*
[2]*Emirates College of Technology, Abu Dhabi, U.A.E.*

Keywords: Semi-Structured Data, Document-oriented Database, Big Data, NoSQL.

Abstract: New business applications require flexibility in data model structure and must support the next generation of web applications and handle complex data types. The performance of processing structured data through a relational database has become incompatible with big data challenges. Nowadays, there is a need to deal with semi-structured data with a flexible schema for different applications. Not only SQL (NoSQL) has been presented to overcome the limitations of relational databases in terms of scale, performance, data model, and distribution system. Also, NoSQL supports semi-structured data and can handle a huge amount of data and provide flexibility in the data schema. But the data models of NoSQL systems are very complex, as there are no tools available to represent a scheme for NoSQL databases. In addition, there is no standard schema for data modelling of document-oriented databases. This study proposes a semi-structured data model for big data (SS-DMBD) that is compatible with a document-oriented database, and also proposes an algorithm for mapping the entity relationship (ER) model to SS-DMBD. A case study is used to evaluate the SS-DMBD and its features. The results show that this model can address most features of semi-structured data.

1 INTRODUCTION

Nowadays, business applications require databases with the ability to support extreme scales and deal with many data formats. Information technology in the healthcare sector is shifting from structure-based data to semi-structured (Wang et al., 2018). Assunção et al. (2015) and Stanescu et al. (2016) discussed the challenges of big data, such as how to deal with increasing data volume and the need for a semi-structured data type to store and handle large amounts of data with a flexible schema.

Assunção et al. (2015) and Siddiqa et al. (2017) discussed the problems of relational databases, which are a challenge in big data handling: how to process and integrate variety and velocity data. Therefore, the Not only SQL (NoSQL) database has been presented as new technology for designing a data model without strict constraints (Wang et al., 2018). NoSQL is capable of accepting all types of structured, semi-structured, and unstructured data and has many features such as a support-distributed system, a flexible schema, and horizontal, scalable, and easy replication (Storey and Song, 2017; Quattrone et al., 2016). Moreover, NoSQL has a different data model concept that is classified according to the storage and retrieval of data, as each model has different ways of designing, storing, and processing data (Storey and Song, 2017). A document-oriented database is designed to manage and store data in document format and collections. A document's contents are encapsulated or encoded in a standard format such as extensible markup language (XML), JavaScript Object Notation (JSON), or Binary JavaScript Object Notation (BJSON) for storing and retrieving the data (Li et al., 2014). Each document has a unique primary key. Also, a document can include different data types, such as complex data structure, nested objects, arrays, and embedded documents (Zhao et al., 2013).

On the other hand, semi-structured data is emerging as one of the best models for handling large amounts of data. Hashem and Ranc (2016) noted that NoSQL distribution supports a schema that will give flexibility in handling and processing semi-structured data.

A semi-structured data format can store data in XML, JSON, or BJSON. Moreover, a document-oriented database stores data in a semi-structured format using the key-value concept. The value of a key can be any data type that gives the database flexibility to store any kind of data (Zhao et al., 2013).

Hamouda, S. and Zainol, Z.
Semi-Structured Data Model for Big Data (SS-DMBD).
DOI: 10.5220/0007957603480356
In *Proceedings of the 8th International Conference on Data Science, Technology and Applications (DATA 2019)*, pages 348-356
ISBN: 978-989-758-377-3

However, there is a lack of semi-structured data models with flexible schemas (Assunção et al., 2015, Mazumdar et al., 2019). These issues and challenges must be addressed by researchers when designing a method or algorithm to retrieve information from a large amount of data (Storey and Song, 2017). Therefore, this study focuses on the design of a semi-structured data model for big data.

The remainder of this paper is structured as follows: the next section reviews existing semi-structured data models; it is followed by the proposed model and algorithm, and presentation of the case study. Finally, the proposed model is evaluated and conclusions are drawn.

2 MODELS OF SEMI-STRUCTURED DATA

A semi-structured data model is used to store different types of data without a formal structure (Strohbach et al., 2016). Numerous properties must be addressed to handle semi-structured data, such as no strict structure, no strict participation/instances, hierarchical structure, non-hierarchical structure, ordering, irregular structure of data, disjunction, self-evolving, mixed content, abstraction, explicit separation of structure and content, partial relationship/participation, heterogeneity, *n*-array relationships, inheritance, reuse potential, constraints, functional dependencies, symmetric relationships, and recursive relationships (Ganguly and Sarkar, 2012). These properties are evaluated in terms of semi-structured data models in Table 1.

Ganguly and Sarkar (2012) found that none of these models can address all of the properties and requirements of the semi-structured data model. The GOOSSDM model can handle most semi-structured requirements, but the integration of semantic web technologies is still problematic with all of these models.

Hashem and Ranc (2016) and Yusof and Man (2017) found that JSON is more compact than XML for semi-structured data because XML has many rules and great complexity in representing a semi-structured data format. JSON is a serialization format, which has schema-less data with many data types, such as string, number, list and array, and nested structure (Florescu and Fourny, 2013). The features of JSON include easy preparation, string array analysis, and suitability for semi-structured data and a cloud database (Mathew and Kumar, 2015). Therefore, JSON plays an important role in representing semi-structured data in a NoSQL database, as it is lightweight and flexible in dealing with formatted, semi-structured data (Storey and Song, 2017).

Table 1: Evaluation of the semantic properties of semi-structured conceptual models (Ganguly and Sarkar, 2012).

Property \ Model	ERX	ORA-SS	XER	EReX	XUML	XSEM	GOOSSDM
No strict structure	✓	✓	✓	-	✓	✓	✓
No strict participation/instances	✓	✓	✓	-	✓	✓	
Hierarchical structure	✓	✓	✓	✓	✓	✓	✓
Non-hierarchical structure		P	X	✓	P	✓	✓
Ordering	✓	✓	✓		P	✓	✓
Irregular structure of data	X		X	✓	✓	✓	✓
Disjunction	X	✓		✓	✓	X	✓
Self-evolving	X	✓			X		✓
Mixed content	X	✓	✓	✓	✓	✓	✓
Abstraction	X			✓			
Explicit separation of structure and content		✓	✓				✓
Partial relationship/participation		✓	✓	✓	✓	✓	✓
Heterogeneous		✓		✓	✓		
N-array relationship		✓			✓		✓
Inheritance				✓	✓		✓
Reuse potential		✓	✓			✓	✓
Constraints	✓	✓		✓	✓	✓	✓
Cardinality		✓					

ERX:Entity relational for XML;ORA-SS:object relationship attribute model for semi-structured data; XER:Extensible ER; EReX:Entity relational extended to XML;XUML:Executable of Unified Modeling Language;XSEM:Conceptual model for XML;GOOSSDM:Graph object-oriented semi-structured data model; ✓=fully supported; X = not fully supported; P=partially supported.

3 PROPOSAL: A SEMI-STRUCTURED DATA MODEL FOR BIG DATA

This study proposes a schema for a semi-structured data model for big data (SS-DMBD) based on the document-oriented data model. This schema is focused on organizing data into semantic collections. Some entities can be grouped into other collections, and documents in the same collection may not have the same fields. The order of the fields is not necessary, and the content of a particular field may differ across documents. The SS-DMBD transforms the conceptual model to a logical model based on the following features of the entity relationship ER model: identifying all entities and attributes, identifying the relationships between entities, resolving all types of relationships, keys (primary, foreign), constraint, and normalization.

3.1 SS-DMBD Components

Many components are required in designing a database based on the data model, to contain the logical and physical data model. This section describes the main components in designing an SS-DMBD:

i. Each strong entity is represented by a collection;
ii. Each weak entity is represented by an embedded document in the strong collection;
iii. Each entity record represents a document;
iv. Each attribute is represented by key-value pairs in which the key represents the attribute and the value the data type of this attribute according to the type of value;
v. The array data type is used to represent multiple values or many documents; and
vi. Embedded and reference documents represent the types of relationships between the collections.

3.2 Features of SS-DMBD

The features of the ER model are used to outline a database schema model. The SS-DMBD improves upon the document-oriented schema of Bhogal and Choksi) 2015 as follows.

3.2.1 Entities

The SS-DMBD database entities include:

Strong entities: A strong entity is a collection of documents represented by a folded corner shape with the entity name written on it.

Weak entities: A weak entity is a document embedded in the strong entity, represented by a folded corner inside the strong entity with the name of the weak entity.

Hierarchical entities: A document is created for each higher level entity with all of its attributes as key values; then, the lower level entities with their attributes are added as an array of documents embedded in the high-level document.

3.2.2 Constraints

The document constraints are the following:

The primary key of the relational schema for each entity is the same for this model. It uniquely identifies each document. The primary key for the document is identified by underlining the relevant attribute.

A foreign key is indicated by adding the primary key of the entity to another entity. The foreign key is represented by a dashed underline of the relevant attribute.

3.2.3 Relationships

The relationships of the document-oriented model based on SS-DMBD are represented as follows:

i. One-to-One (1:1) Relationship: This can be handled in two ways: using an embedded document or using the reference document. If one side of the relationship has datasets not exceeding 16 MB or has less than tens of thousands/hundreds of thousands/millions records, and there are no relationships with other entities, then an embedded document is used to handle this relationship. If more relationships exist or both sides have datasets including more than tens of thousands/hundreds of thousands/millions of records, then this relationship should be represented using the reference document.

The embedded document is represented by a folded upper right corner and is marked with the type of relationship.

The reference document is used to represent the relationships between two collections and is described by a line between them.

ii. 1:N Relationship: This relationship can be described using the embedded document or reference document depending on the size of datasets for the N side. If N is small (i.e., less than tens of thousands/hundreds of thousands/millions records, based on the assumption that documents are not large), the upper left corner of the embedded

document is folded and marked with the type of the relationship. If N is large (i.e., more than tens of thousands/hundreds of thousands/millions of records), then the relationship is described by the reference document.

iii. M:M Relationship: This type of relationship can be handled by creating two collections, then storing a list of related documents with links to the other collections as a list of array elements in other documents and vice versa. It is represented using an array of data type with the embedded documents. Square brackets [ ] are used to identify the array, and each element of the array is a document stored in the field "relationship name or combination of both collections' names."

iv. Unary Relationship: A unary relationship is described by normal key-value pairs in the samedocument, and K is the name of the unary relationship.

4 MAPPING ER SCHEMA TO SS-DMBD

The components of ER schemas are entities, attributes, and relationships. SS-DMBD uses E to represent an entity, and a series of entities will be Ei….En (i=1 to n). The number of attributes will be represented by Aj….An (j=1 to n), and R is used to represent the type of relationships (1:1 or 1:N or M:M).

4.1 SS-DMBD Specifications

ER specifications are converted to the SS-DMBD specifications as shown in Table 2.

Table 2: Specifications of SS-DMBD.

Database properties	ER model	SS-DMBD notion	Description
Entity (E)	Strong entity	C	Create a new collection
	Weak entity	WC	Embed weak entity into a strong collection
	Hierarchical entities	HC{LC1,…,LCi} HC: Higher collection LC: lower collection i: number of lower collections	Create a document for all higher level entities with all the attributes as key values. Add the lower level entities with their attributes as an array of embedded documents for the high-level document. HC{k1..kn, LC[{k1..kij}]……}
Attribute	Attributes	{K1,…,Ki}	The attributes of each entity are described using K; they are listed in brackets as documents separated by commas.
	Multi-valued attribute	MV[K1,…,Ki]	The multi-valued attribute is described by the name of this attribute with an array data type, and Vi represents all of the multi-values.
Relationships	Relationship types	E_m	The embedded model applies between two entities.
		R_f	The reference model applies between two entities.

Algorithm for mapping the ER schema to SS-DMBD

Input: ER schema
Output: SS-DMBD

1: BEGIN.
2: for each strong entity
3: create new collection $C_{i(i=1…n).}$
4: for each weak entity $WC_{i(i=1…n)}$ do
5: create embedded documents belonging in the strong entity (WC_i $E_m \subseteq C_i$).
6: end for.
7: for each multi-value attribute "MVi" do
8: store multi-values as array data type belonging in the strong entity $\forall$ $MV_{i(i=1…n)}$ []$\subseteq$ Entity.
9: end for.
10: for each 1:1 relationship between two entities (Entity1 and Entity2) do
11: If Entity1 data set size is less than 16 MB or it has less than tens of thousands/hundreds of thousands/millions of records and no other relationship with other entity.
12: Entity1 store as an embedded document into Entity2 (Entity 1 $E_m \subseteq$ Entity 2).
13: else

14: apply reference document between Entity 1 and Entity 2 (Entity 1 $R_f \subseteq$ Entity 2).
15: end if.
16: end for.
17: for each 1:N relationship between two entities (Entity 1 and Entity 2) do
18: if N data set size is less than 16 MB or it has less than tens of thousands/hundreds of thousands/millions of records then
19: N side store as an embedded document into 1 side (Entity2$_{(N)}$ Entity $_m \subseteq$ E Entity1$_{(1)}$).
20: else
21: apply reference document between Entity1 and Entity2 (Entity 1 $R_f \subseteq$ Entity2)
22: end if.
23: end for.
24: for each M:M relationship between two entities (Entity1 and Entity2) do
25: for Entity1 side do
26: create array data type into an embedded document
store the primary key of Entity2 with other related attributes.
27: Update Entity1 with the embedded document of Entity2 (Entity2 :{[E_m]} $\subseteq$ Entity1).
28: end for.
29: for E2 do
30: create array data type in an embedded document
31: store the primary key of Entity1 with other related attributes.
32: update Entity2 with the embedded document of Entity1 (Entity1 :{[Em]} $\subseteq$ Entity2).
33: end for.
34: end.

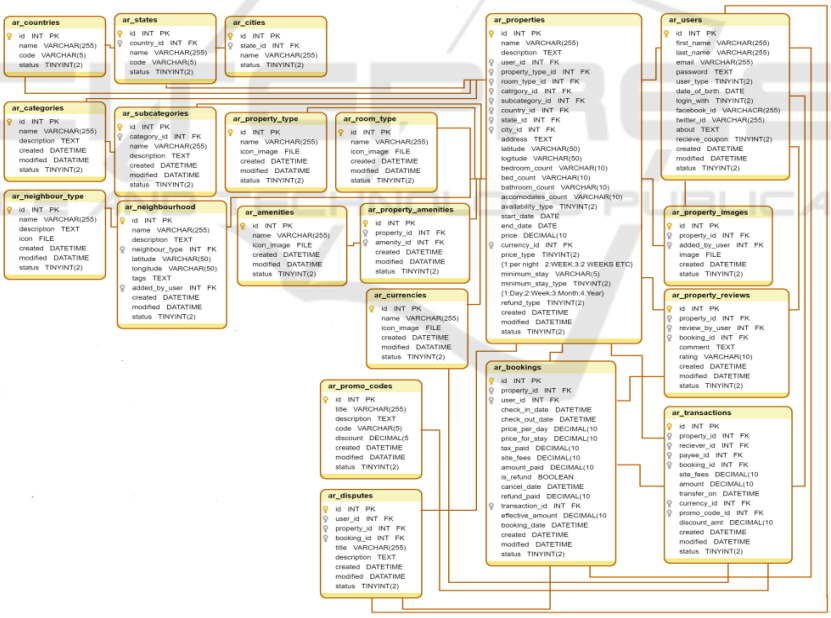

Figure 1: Airbnb schema in relational database (https://creately.com/app/?tempID=idbbu9op2#).

5 CASE STUDY: AIRBNB, INC.

Airbnb, Inc. is one of the most effective travel accommodation and apartment-providing services worldwide. According to Quattrone et al., (2016), Airbnb need to changing their business requirements from time to time.

5.1 Airbnb Schema in Relational Database

Figure 1 presents a schema for the large company Airbnb in a relational database.

5.2 Airbnb Schema by SS-DMBD

A mapping algorithm was applied to map the Airbnb schema from the relational database to SS-DMBD; the output is shown in Figure 2.

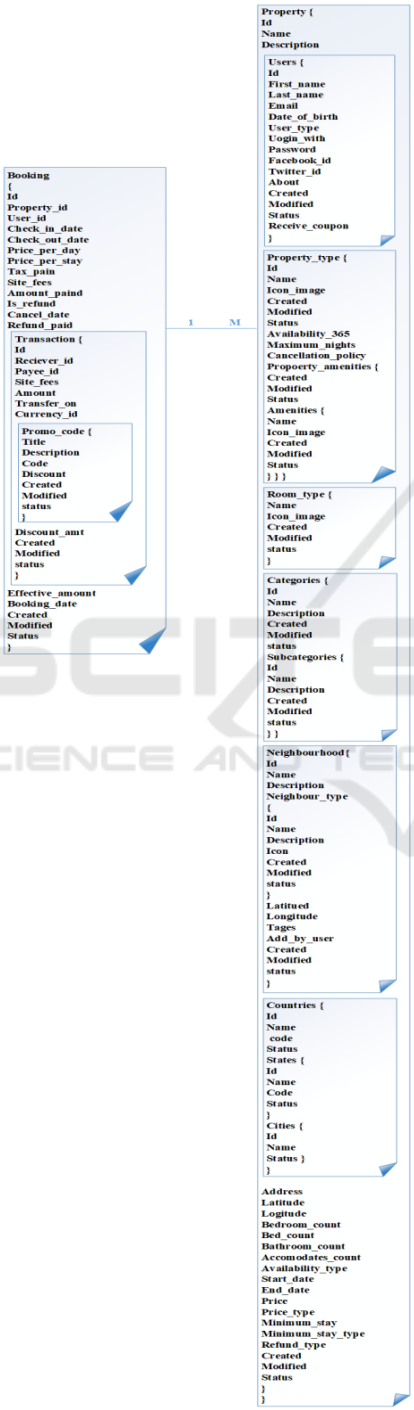

Figure 2: Airbnb schema by a semi-structured data model for big data (SS-DMBD).

6 EVALUATION OF SS-DMBD FEATURES WITH SEMI-STRUCTURED PROPERTIES

SS-DMBD covers most semantic features of semi-structured data, as follows.

6.1 No Strict Structure

SS-DMBD has no strict formatting, which indicates that similar attributes in each document are irrelevant and new attributes can be added at any time without any strict structure.

6.2 Hierarchical Structure

SS-DMBD can support tree data structures of hierarchical relationships. The concept of embedding a document into a collection considers the hierarchical structure between two collections, and each collection may have many embedded documents.

6.3 Non-hierarchical Structure

The document of parent references stores each tree node, and each tree node stores the ID of a reference document. Thus, each collection may include many references to other collections with no hierarchical structure.

6.4 Ordering

The concept of the document-oriented database model is a free schema—that is, the order of key values in the document depends on how they are inserted. Each document may have a different ordering of fields.

6.5 Irregular Data Structure

SS-DMBD stores data in a semi-structured format that is, data are stored as key values. The value of the key can be any type of data, and pairs of key values are stored in documents. Given that each document has a flexible schema; data can be stored in an irregular structure.

6.6 Disjunction

Disjunction is represented by an embedded document. The main document ID is associated with

the embedded document's ID, and the embedded ID can access all the embedded fields.

6.7 Self-evolution

Self-evolution considers as the main concept of the documents oriented model of because it does not have a fixed schema. Therefore, each document has a self-description, which allows each key to be self-described.

6.8 Mixed Content

SS-DMBD allows each key to have different contents in each document, and these contents can be blended in the same document without a structure. The concept of key values is to accept any kind of data without constraint and without defining the type of key, because the constraint of value will be responsible for the application.

6.9 Abstraction

SS-DMBD hides the complexity of data. Given that the content in each key is not important, a key can be accessed without any details about the value type.

6.10 Explicit Separation of Structure and Content

The logical structure of documents is represented in a separate hierarchy by considering the content of the key-value series.

6.11 Partial Relationship/Participation

In the data model of a document, a relationship can be represented using reference and embedded documents to identify the parent and child of each document. The main collection includes the parent and embedded documents (which are considered the children). The reference model between collections represents participation.

6.12 N-array Relationship

SS-DMBD represents the many-to-many relationship and multi-value attributes in array values and makes better use of this feature than previous models did. Embedded and reference documents represent many-to-many relationships.

6.13 Inheritance

SS-DMBD supports inheritance through the embedded document. The main document can describe the common properties, and the embedded document can be represented by the sub-properties of the main document data. Therefore, SS-DMBD can describe inheritance.

6.14 Reuse Potential

The reference linking of the relationship between two documents describes the reuse potential. For example, the relationship between collections can be represented by a reference document. Thus, SS-DMBD can connect the collection with other collections through the reference concept.

6.15 Constraints

SS-DMBD is a flexible schema, meaning it has no constraints when dealing with different data types or when documents may have different fields depending on the system. Moreover, if a constraint needs to be applied, it will be implemented by a programming application.

6.16 Cardinality

An embedded document can represent the relationships between the document and collections. The reference document can represent the relationships between the collections. SS-DMBD supports cardinality features, such as the relational database that keeps relationships between tables through embedded and reference documents.

6.17 Functional Dependencies

The data are organized into key values. Each key is used to store a value, which can be determined by the key. Each document also has a primary key, which is used to identify all document keys. Therefore, all document fields are functionally dependent on the primary key document.

6.18 Symmetric Relationship

Each value in the document can be related to the key, and each key is symmetrically related to the value.

6.19 Recursive Relationship

This feature can be described through the

relationships between documents. A unary relationship is stored in the related collection, which can be described as a recursive relationship.

6.20 Flexible Schema

This new feature is required for big data not covered by any semi-structured models or a relational database, and for new business requirements and changes. In SS-DMBD, the schema can be changed at any time. SS-DMBD allows adding any field in any document without constraint and allows each document to have different numbers of fields. It also allows changing the relationships before or after implementation.

6.21 Time Stamp

This feature is required for semi-structured data. Real-time applications need a way to evolve with time. Time-stamp data type can be better and more efficient than date-time data type. SS-DMBD provides time-stamp data for each document that it supports.

To summarize, in a relational database, a new field should be added to change its schema, but the empty field will cause inefficiencies in performance. SS-DMBD addresses this issue by allowing adding or altering data in any structure without changing the database schema. SS-DMBD allows the application to use the required data and ignore unrequired data. The flexible schema and time stamp are fully supported by SS-DMBD, unlike previous semi-structured models.

7 CONCLUSION

A semi-structured data model was designed to be compatible with a document-oriented database. Also, an algorithm was proposed to map the ER model to SS-DMBD. This algorithm can be used to convert any relational database schema to a document-oriented database schema. Furthermore, semi-structured data can be formatted in a document in a way that is more useful than a table when a large amount of data is available. The proposed model provides features for the conceptual representation of a document-oriented database. For example, it presents a flexible schema by allowing the application to change or update business requirements over time, it allows collecting data of different types from different sources, and it represents relationships as embedded and reference documents. The study can be extended to migrate a relational database to a document-oriented database.

REFERENCES

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A. & Buyya, R. 2015. Big Data Computing and Clouds: Trends and Future Directions. *Journal of Parallel and Distributed Computing*, 79, 3-15.

Bhogal, J. & Choksi, I. Handling Big Data Using NoSQL. Advanced Information Networking and Applications Workshops (WAINA), *2015 IEEE 29th International Conference On*, 2015. IEEE, 393-398.

Feng, W., Gu, P., Zhang, C. & Zhou, K. Transforming UML Class Diagram Into Cassandra Data Model With Annotations. 2015 IEEE International Conference on Smart City/Socialcom/Sustaincom (SmartCity), 2015. IEEE, 798-805.

Florescu, D. & Fourny, G. 2013. JSONiq: The History of a Query Language. *IEEE Internet Computing*, 17, 86-90.

Ganguly, R. & Sarkar, A. 2012. Evaluations of Conceptual Models for Semi-Structured Database System. *International Journal of Computer Applications*, 50.

Hashem, H. & Ranc, D. Evaluating NoSQL Document Oriented Data Model. *Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on*, 2016. IEEE, 51-56.

Li, X., Ma, Z. & Chen, H. QODM: A Query-Oriented Data Modeling Approach for NoSQL Databases. *Advanced Research and Technology in Industry Applications (WARTIA), 2014 IEEE Workshop on*, 2014. IEEE, 338-345.

Mathew, A. B. & Kumar, S. M. Analysis of Data Management and Query Handling in Social Networks using NoSQL Databases. Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on, 2015. IEEE, 800-806.

Mazumdar, S., Seybold, D., Kritikos, K. & Verginadis, Y. 2019. A Survey on Data Storage and Placement Methodologies for Cloud-Big Data Ecosystem. *Journal of Big Data*, 6, 15.

Quattrone, G., Proserpio, D., Quercia, D., Capra, L. & Musolesi, M. Who Benefits From The Sharing Economy of Airbnb? *Proceedings of the 25th International Conference on World Wide Web, 2016. International World Wide Web Conferences Steering Committee*, 1385-1394.

Siddiqa, A., Karim, A. & Gani, A. 2017. Big Data Storage Technologies: A Survey. *Frontiers of Information Technology & Electronic Engineering*, 18, 1040-1070.

Stanescu, L., Brezovan, M. & Burdescu, D. D. Automatic Mapping of MySQL Databases to NoSQL MongoDB. *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*, 2016. IEEE, 837-840.

Storey, V. C. & Song, I.-Y. 2017. Big Data Technologies and Management: What Conceptual Modeling Can

Do. *Data & Knowledge Engineering*, 108, 50-67.
Strohbach, M., Daubert, J., Ravkin, H. & Lischka, M. 2016. Big Data Storage. *New Horizons for a Data-driven Economy*. Springer, Cham.
Wang, Y., Kung, L. & Byrd, T. A. 2018. Big Data Analytics: Understanding its Capabilities and Potential Benefits for Healthcare Organizations. *Technological Forecasting and Social Change*, 126, 3-13.
Yusof, M. K. & Man, M. 2017. Efficiency of JSON for Data Retrieval in Big Data. *Indonesian Journal of Electrical Engineering and Computer Science*, 7, 250-262.
Zhao, G., Huang, W., Liang, S. & Tang, Y. Modeling Mongodb with Relational Model. *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*, 2013. IEEE, 115-121.