Normalization and Denormalization for a Document-Oriented Data

Shady Hamouda[1], Mohammed Anbar[2] and Omar Elejla[3]
[1]*Department of Business Information Technology, Liwa College of Technology, Abu Dhabi 51133, U.A.E.*
[2]*National Advanced IPv6 (NAv6) Centre, University Sains Malaysia, USM, Penang 11800, Malaysia*
[3]*Department of Computer Science, Al-Aqsa University, Gaza 4051, Palestine*

Keywords: Normalization, Embedded, Reference Document, NoSQL, Document-Oriented Database.

Abstract: Recently, the challenge of the increasing volume of data has led to the presentation of the "not only structured query language (NoSQL) database". One of the most powerful types of NoSQL databases is the document-oriented database that supports a flexible schema. Normalization of the data model is one of the important research issues and there are no standard principles of normalization in the document-oriented database. Handling relationships based on normalization and denormalization has not been considered in document-oriented databases despite its importance probably because it is not recommended in creating a collection for each entity or using a reference document for all because of the need to execute a complex joint operation. Recently, many researchers have migrated from relational databases to document-oriented databases. However, their migration methods are facing issues; first is no method to normalize or de-normalize data to implement the embedded and reference document. Second, migration from a relational database to a document-oriented database does not consider how to handle various types of relationships based on normalization and de-normalization. This study proposed a way to deal with migration problems by enhancing transformation rules to map entity relational schema to document-based data schema based on normalization and denormalization data. The results of this study show that the dataset size determines whether reference or embedded documents should be used for migration.

1 INTRODUCTION

With the rise of big data, a relational database has been unable to fit the dimensions of big data, especially data velocity and variety (Abourezq & Idrissi, 2016). According to Younas (2019), storing and managing big data requires new data models and technologies. These issues and challenges have led to the development of a Not only SQL (NoSQL) database to overcome the limitations of the relational database, such as designing a schema without strict constraints (Hashem & Ranc, 2016; Truică, Apostol, Darmont, & Pedersen, 2021).

As Abdelhedi, Brahim, Atigui, and Zurfluh (2018) mentioned, a document-oriented database has proven to be the most adapted solution that supports a larger volume of data and provides a flexible schema. Moreover, Younas (2019) found that the document-oriented database can be suitable for high development productivity and low maintenance cost of modern Web 2.0 applications for two main reasons: First, these applications have a constant evolution of data schema and benefit from the flexible schema of the document-oriented database; second, Web 2.0 applications support data models such as JSON with tight integration with popular programming languages such as Python, JavaScript, and Ruby (Bathla, Rani, & Aggarwal, 2018).

Normalization and de-normalization data in a document-oriented database differ from those in the relational database. However, it should provide rules to implement normalization and de-normalization data in the document-oriented database. Therefore, this is one of the most important areas addressed in the current research, as it is critical to understanding the process of normalization and de-normalization. At the same time, it can also affect database performance and storage space (Mehmood *et al.*, 2017; González-Aparicio *et al.*, 2017).

Addressing relationships based on embedded and reference documents in document-oriented databases must be considered (Mehmood *et al.*, 2017)—although important, creating a collection for each entity or using a reference document for all entities is not recommended because doing so would require the

Hamouda, S., Anbar, M. and Elejla, O.
Normalization and Denormalization for a Document-Oriented Data.
DOI: 10.5220/0011868100003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 972-979
ISBN: 978-989-758-623-1; ISSN: 2184-433X

performance of complex joint processes. Also, storing all entities as embedded documents in a single collection is not useful because it will produce a large amount of unnecessary and inconsistent data. Additionally, all data would be uploaded when updated, thereby reducing performance. Therefore, document-oriented databases should be designed using embedded and reference document technologies to improve synchronization when updating redundant data (Atzeni *et al*., 2016). In addition to that, the study by Imam *et al.* (2018) mentioned issues that still need to be addressed to implement a document-oriented database, such as how to represent one-to-many relationships in document-oriented databases, as well as how and when to use reference documents instead of embedding documents. However, Oliveira, Oliveira, and Alturas (2018) found that there are no investigations to understand the migration process or the methodology of migrating from a relational database to a document-oriented database.

The aim of this study is to facilitate the process of transformation of the relational database schema to a document-oriented data schema through two concepts: the first is clarifying the embedded document (de-normalization) relationships by storing the sub-document into a super-document collection; the second is using the reference document to normalize the relationship by linking the collections with a foreign key.

2 BACKGROUNDS AND RELATED WORK

The previous model addresses how to transform a strong entity by creating a new collection and transform the relationships of one-to-one by embedded document without taking into consideration the size of datasets and not addressing how to apply the embedded and reference document for other relationship types such as one-to-many, many-to-many, and unary relationship. Additionally, the weak entity has been transformed by using new collections, while it should belong to the strong entity as an embedded document to avoid many join operations between many collections.

Many researchers have proposed methods to migrate relational databases to the document-oriented database (Corbellini, Mateos, Zunino, Godoy, & Schiaffino, 2017; El Alami & Bahaj, 2016; Goyal, Swaminathan, Pande, & Attar, 2016; Győrödi, Győrödi, Pecherle, & Olah, 2015; Hanine, Bendarag, & Boutkhoum, 2016; Imam, Basri, Ahmad, Watada, & González-Aparicio, 2018; Karnitis & Arnicans, 2015; Mason, 2015; Stanescu, Brezovan, & Burdescu, 2016, 2017; Yoon, Jeong, Kang, & Lee, 2016). For instance, El Alami and Bahaj (2016); Hanine et al. (2016); Mason (2015); Stanescu et al. (2016, 2017) have focused on migrating a relational database to a document-oriented database based on the concept of embedded and reference documents.

However, these migration methods are facing various issues; the first issue is that no specification can be recognized to define a schema for a document-oriented database due to the various ways of storage, management, and implementation in document-oriented databases (Goyal *et al*., 2016). The lack of presenting a schema led to present many challenges and complex problems in migration because designing a schema for the document-oriented database is important for defining the principles and overcoming the issues of relationship types for document-oriented databases (Truică, Apostol, Darmont, & Pedersen, 2021). Also, it may lead to incorrect or inappropriate schema design, especially when handling relationships based on normalizing and de-normalizing data. For instance, the method of Stanescu *et al*. (2017), did not properly migrate all the database properties especially, the multi-values, weak entity, and relationship types. Some migration result is an embedded document while they should be migrated by using an array data type as it contains one field with many values. In addition, if there is any table refereed by more than two other tables and has more than one foreign key. These cases were missing in the Stanescu *et al*. (2017) algorithm.

Additionally, there is no technique method to normalize or de-normalize data to implement the embedded and reference document for handling the various types of relationships (Hanine *et al*., 2016; Mehmood *et al*., 2017). According to Mehmood, Culmone, and Mostarda (2017), normalization (reference document) and de-normalization (embedded document) are the two techniques that must be considered when designing a schema. These techniques can affect the performance and storage effectively as the databases grow rapidly. González-Aparicio *et al*. (2017) observed that the normalization of the data model is one of the important research issues and there are no standard principles of normalization in the document-oriented database.

The transformation rules of previous work have mapped the relational database schema to the document-oriented database directly without considering any specification (Varga et al., 2016; Mehmood et al., 2017; Mior et al., 2017; Imam et al., 2018; Stanescu et al., 2017). For instance, study by Stanescu et al. (2017) migrated the relational database to a document-oriented database (MongoDB) based on the number of foreign keys in each table as well as the number of tables referring to that table. However, these studies neither normalized nor de-normalized data for a document-oriented database and did not propose a method to explain how to implement embedded and reference documents to represent the relationships, even though Mehmood et al. (2017) mentioned that the quality of the schema can be assessed through normalized and de-normalized data.

The second issue is the transformation rules that are needed to normalize and de-normalize data for handling the relationship types based on embedded and reference documents (Jouini & Engineering, 2022). This issue has not been considered in document-oriented databases despite its importance probably because it is not recommended for creating a collection for each entity or using a reference document for all because of the need to execute a complex joint operation. Furthermore, storing all the entities as embedded documents in one collection is not beneficial because it will cause many redundant and inconsistent data (Atzeni, Bugiotti, Cabibbo, & Torlone, 2016).

Finally, the migration from a relational database to a document-oriented database does not consider all the database properties, especially on how to handle various types of relationships. Because migration without any specification or methodology to normalize and de-normalize the various types of relationships will cause incorrect migration (Colombo & Ferrari, 2019; El Alami & Bahaj, 2016; Győrödi et al., 2015; Hanine et al., 2016; Stanescu et al., 2017).

3 PROPOSED METHOD

To enhance migration process performance, this study enhances the mapping of ER schema to Document-oriented data schema by enhancing the rules of mapping the ER specifications such as weak entity, hierarchical entity, composite attribute, multi-valued attribute, derived attribute, attribute relationship, and constraint. Also, handling the relationship types (1:1, 1:M, M: M, Unary) using embedded and reference documents. These enhancements points are converted to Transformation Rules (TR.)

3.1 Enhancement of the Transformation Rules

The rules developed by Stanescu et al. (2017) are based on checking the number of foreign keys in each table as well as the number of the table referring to that table. However, these rules do not cover all database properties and cannot be applied to a complex schema. Additionally, the rules do not consider how to handle the relationship types based on embedded and reference documents. Therefore, this study has enhanced TRs based on embedded and reference documents.

3.2 Embedded and Reference Documents

This study aims to facilitate the process of transformation of the relational database schema to a document-oriented data schema through two concepts: the first is clarifying the embedded document (de-normalization) relationships by storing the sub-document into a super-document collection; the second is using the reference document to normalize the relationship by linking the collections with a foreign key.

One of the challenges that migration methods face is when the document-oriented database is not supported in the joint operation. Consequently, most researchers and specialists look for alternative joint processes and utilize the embedded document as a methodology between collections. However, the embedded document is not suitable for most large applications. At present, the document-oriented database (MongoDB) is supporting a joint process. Thus, it requires a method for choosing between the embedded and reference documents.

3.2.1 Embedded Document

The embedded document refers to the embedding of related data (key-value, document, collection) into the document so it can define the de-normalization concept by storing all data into one single document, which leads to duplicate or inconsistent data. The study represents the relationship between entities by storing all the data in one collection with related documents as an embedded document.

The embedded documents (Figure 1) cannot be used to reduce the joint operation between collections; at the same time, it can cause duplicated and inconsistent data. It can also cause bad performance in the updated document, as it needs to load all the collections.

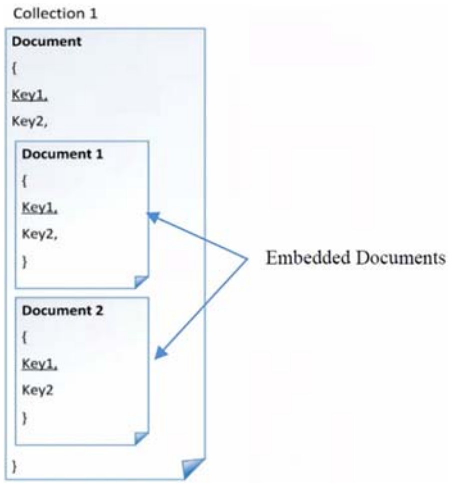

Figure 1: Embedded documents.

3.2.2 Reference Document

The reference document refers to the application of the normalization concept by storing the data in multiple collections with references between those collections using the concept of the foreign key to support a joint operation between the database collections.

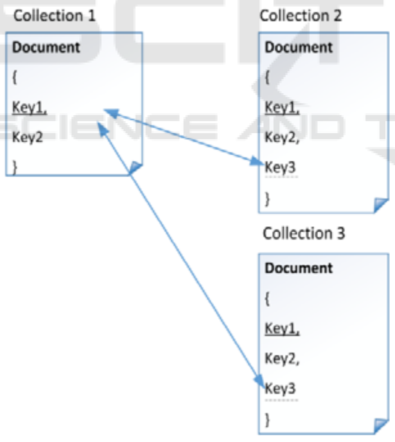

Figure 2: Reference document.

The concept of the reference document (Figure 2) is similar to that of the relational database, which means storing the entity in different collections and making the relationships by using a foreign key. This concept can be used in the case of reducing the embedded documents and it can support flexibility in storing the growing amount of data. However, it is not preferable to have many collections in the database because doing so will require a complex joint operation.

3.3 Proposed Transformation Rules

The ER schema comprises the following components: entities, attributes, and relationships.

This study is representing the entity by E and a series of entities as $E_i \ldots E_n$ $_{(i=1 \text{ to } n)}$, while the number of attributes was represented by using $A_j \ldots A_{n(j=1 \text{ to } n)}$, and R is used to represent the type of relationships: (1:1), (1:N), and (M: M). Table 1 shows the notations used to map the ER schema to DOD_S.

The presented TRs describe how to map the ER schema to the DOD_S. This study proposed six transformation rules. These rules take the ER schema as input and map the DOD_S as output.

```
 Rule 1:  For each strong entity
$Ei_{(i=1…n)}$
 create a new collection $Ci_{(i=1….n)}$ ,
where n= number of collections
Rule 2:  For each weak entity $E_{(weak)i}$
$_{(i=1…n)}$
Create $E_{(weak)}$  $_{(i=1…n)}$ as embedded
documents belonging to the strong
entity
 $\forall$ ($E_{(weak)i}$ embedded $\subseteq E_i$)
Rule 3: For each multi-value attribute
$A_{(Multivalue)i}$
store multi-values as array data type
belonging to the strong entity
      $\forall$ $A_{(Multivalue)i}$ $_{(i=1…n)}$ [ ]$\subseteq$ Ei
Rule 4: For each (1:1) relationship
between two entities (Ei R Ej)
If Ei dataset's size is less than 16 MB
and no other relationship exists with
another entity
 Ei stored as an embedded document into
 Ej  (Ei embedded $\subseteq$ Ej)
 else
  apply reference document between Ei
  and Ej (Ei reference $\subseteq$ Ej)
Rule 5: For each (1:N) relationship
between two entities (Ei R Ej)
 if N dataset's size is less than 16 MB
records then
 N side entity stored as an embedded
document into 1 side entity ($Ej_{(N)}$
embedded         $\subseteq$ $Ei_{(1)}$)
   else
apply reference document between Ei and
Ej (Ei reference $\subseteq$ Ej )
Rule 6: For each (M: M) relationship
between two entities (Ei R Ej)
  i.  store the primary key and related
  key-value of E1 as an embedded
  document into E2 as (Ei :{[embedded]}
  $\subseteq$ Ej).
      ii.   store the primary key and
      related key-value of E2 as an
```

```
    embedded document into E1 as (Ei
    :{[embedded]} ⊆ Ei).
    iii.  apply embedded document
    between Ei and Ej as Ej  (Ej
    :{[embedded]} ⊆ Ei)
    iv.   apply reference document
    between Ei and Ej as ( Ei
    reference ⊆ Ej )
Rule 7: For each (unary) relationship
for Ei entity
    i.    store the primary key of Ei
    with a different name as a foreign
    key into E1 as (K ⊆ Ei).
```

The TRs are used to map the ER schema to the DOD_S. The input of these rules is the ER schema. The first step is to transform the strong entity by creating a new collection with all the data for each strong entity. The second step is to transform the weak entity by creating an embedded document for the weak entity and data into the strong collection. Each multi-value attribute will be stored as an array data type with all the values belonging to that entity.

The relationships are transformed as follows: Relationship (1:1): This relationship is required to determine the volume of each relationship side. If the dataset of one side of the relationship includes small data (i.e., less than tens of thousands/hundreds of thousands/millions of records) or the data size of one side does not exceed 16 MB (the maximum size of the document), then this relationship will transform into an embedded document that will be stored in the relevant collection. If there are more relationships or both cardinalities have large datasets (i.e., more than tens of thousands/hundreds of thousands/millions of records) or the data size of one side exceeds 16 MB, then the relationship must be represented by using the reference document, which will be stored in a different collection.

(1: N) relationship: This relationship is required to determine the volume of the N side. If the N dataset's size is more than 16 MB or the N side is large (i.e., more than tens of thousands/hundreds of thousands/millions of records), then this relationship will be represented through the reference document by a separate collection with the primary key and related attributes of both sides. Otherwise, if the N side is small (the N dataset's size does not exceed 16 MB or the N side has fewer than tens of thousands/hundreds of thousands/millions of records), then it will store the primary key with related data into a set of embedded documents.

M: M relationship: This type of relationship is transformed by using both embedded and reference documents. In the first side of the relationship entity, an array data type and embedded document are created, that contains the primary key of the second entity with other related attributes. At the same time, on the second side of the relationship entity, an array data type and embedded document are created, that contains the primary key of the first entity.

Unary relationship: The attribute of the unary relationship will be stored in the related document. Then, all keys with a null value will be removed, as this form contains a flexible schema, meaning it can remove or add any key value from the document.

These transformation rules map all ER schema to DOD_S and can thereby be used as a strategy for the migration of a relational database to a document-oriented database.

3.4 Case Study: W3school Schema

The third case study is the schema of the W3school website (http://www.w3schools.com/).The W3school schema, presented in Figure 3, can be described as the `PRODUCT` has `CATEGORIES` and `SUPPLIERS`, and `ORDER` has `ORDERDETAILS` and `SHIPPERS` to the `CUSTOMER` through the `EMPLOYEES`.

Table 1: Symbols and type of notations.

Model	Type	Notations	Descriptions
ER Schema	Strong entity	Ei	Ei….n (i=1 to n)
	Attribute	A	Ai….An (i=1 and n number of attributes)
	Relationship	R	Relationships can be one-to-one (1:1), one-to-many (1:M), or many-to-many (M:M)
TRs	Collection	C	
	Key-value	K	
	Embedded document	embedded	The embedded model applies between two entities
	Reference document	reference	The reference model applies to two entities
	Array	[ ]	Array data type

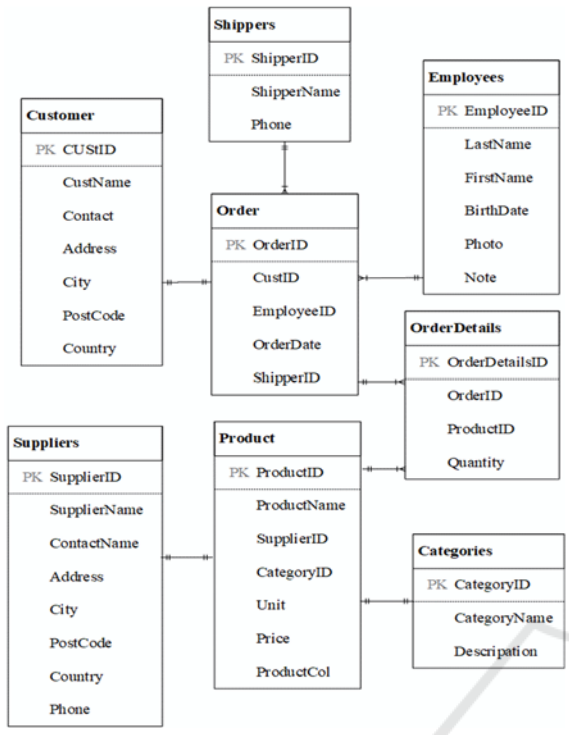

Figure 3: ER schema for W3schools (Rocha, Vale, Cirilo, Barbosa, & Mourão, 2015).

The TRs were applied to the schema shown in Figure 3, and the output of this schema is shown as follows:

i) Created new collections for the main strong entity, which are PRODUCT, ORDER, and EMPLOYEES.
ii) Mapped the relationship between PRODUCT and CATEGORY by storing CATEGORIES as embedded documents in the PRODUCT collection. Also, mapped the relationship between PRODUCT and SUPPLIERS by store SUPPLIERS as embedded documents in the PRODUCT collection
iii) Mapped the relationship between ORDER and ORDERDETAILS by creating an embedded document for ORDERDETAILS in the ORDER collection. Also, the relationship between ORDER and SHIPPERS was mapped to create embedded documents for SHIPPERS in the ORDER collection.
iv) Mapped the relationship between ORDERS and CUSTOMERS by creating an embedded document for CUSTOMERS in the ORDERS collection.

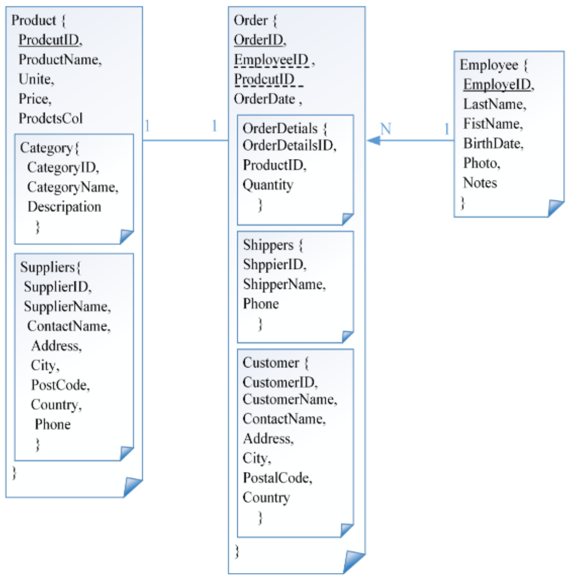

Figure 4: The DODS for W3schools.

As Figure 4 shows, the ER schema first depicted in Figure 3 has been mapped in its entirety without missing any specification. The DODS of Figure 4 contains three collections PRODUCT, ORDER, and EMPLOYEE, as the PRODUCT collection contains CATEGORY and SUPPLIERS entities as embedded documents based on the TRs. Also, the ORDER collection contains ORDERDETAILS, SHIPPERS, and CUSTOMERS entities as an embedded document based on the TRs.

Based on the case study, this study performed two evaluations. In the first evaluation, dataset 3 has 100,000 orders assigned to the order collection using the embedded document. The second evaluation used the reference document by adding the primary key of customer collection in dataset as a foreign key in order collection. Then, study addresses the 10 queries used to test the performance of the embedded and reference documents through database operations. The result and the execution time of each query are presented in the following sections.

The evaluation performed the previous queries (1–10) on MongoDB (document-oriented database) using W3school datasets to determine the performance of embedded and reference documents.

Figure 5 shows the query execution times of the embedded and reference documents based on the proposed method that performed queries 1 to 10. The performance of the reference document was better than that of the embedded document. Therefore, applying the embedded and reference documents

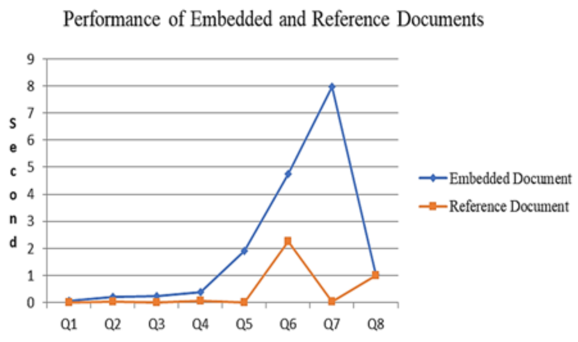

Figure 5: Execution time for applied queries of embedded and reference documents based on dataset.

when migrating the relationship types will depend on the size of the dataset. If the dataset on the relationship side is large (i.e., more than tens of thousands/hundreds of thousands/millions of records, or exceeding 16 MB), then the reference document should be selected; otherwise, the embedded document is preferable.

4 CONCLUSIONS

This study enhances the transformation rules presented to enhance migration process performance that covers all database properties. These transformation rules can be used to map any relational database schema to a document-oriented database. Also, it can overcome the issues in handling the relationships of a complex database and can be used to implement normalization and de-normalization data in a document-oriented database.

The results of this study show that the dataset size determines whether reference or embedded documents should be used for migration. The embedded document is used in case of small dataset (i.e., tens of thousands/hundreds of thousands/millions of records, or the data size of one side did not exceed 16 MB, which is the document size used in MongoDB). By contrast, the reference document is used when the dataset size is large (i.e. tens of thousands/hundreds of thousands/millions of records, or the data size of one side exceeding 16 MB). Reference (normalized) and embedded (de-normalized) documents are important variables for the designed schema and migrate a relational database to a document-oriented database using these transformation rules.

In future work, this study will extend to assessing the performance of embedded and reference documents for a document-oriented data schema based on the proposed method.

REFERENCES

Abdelhedi, F., Brahim, A. A., Atigui, F., & Zurfluh, G. (2018). *Towards Automatic Generation of NoSQL Document-Oriented Models.* Paper presented at the Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA).

Abourezq, M., & Idrissi, A. (2016). Database-as-a-service for big data: An overview. *International Journal of Advanced Computer Science and Applications (IJACSA), 7*(1).

Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2016). Data modeling in the NoSQL world. *Computer Standards & Interfaces.*

Bathla, G., Rani, R., & Aggarwal, H. (2018). Comparative study of NoSQL databases for big data storage. *International Journal of Engineering & Technology, 7*(26), 83.

Chouder, M. L., Rizzi, S., & Chalal, R. (2017). *Enabling Self-Service BI on Document Stores.* Paper presented at the EDBT/ICDT Workshops.

Colombo, P., & Ferrari, E. (2019). Access control technologies for Big Data management systems: literature review and future trends. *Cybersecurity, 2*(1), 3.

Corbellini, A., Mateos, C., Zunino, A., Godoy, D., & Schiaffino, S. (2017). Persisting big-data: The NoSQL landscape. *Information Systems, 63*, 1-23.

El Alami, A., & Bahaj, M. (2016). *Migration of a relational databases to NoSQL: The way forward.* Paper presented at the Multimedia Computing and Systems (ICMCS), 2016 5th International Conference on.

Goyal, A., Swaminathan, A., Pande, R., & Attar, V. (2016). *Cross platform (RDBMS to NoSQL) database validation tool using bloom filter.* Paper presented at the Recent Trends in Information Technology (ICRTIT), 2016 International Conference on.

Hanine, M., Bendarag, A., & Boutkhoum, O. (2016). Data Migration Methodology from Relational to NoSQL Databases. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 9(12), 2369-2373.

Hashem, H., & Ranc, D. (2016). *Evaluating NoSQL document oriented data model.* Paper presented at the Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on.

Imam, A. A., Basri, S., Ahmad, R., Watada, J., & González-Aparicio, M. T. (2018). Automatic schema suggestion model for NoSQL document-stores databases. *Journal of Big Data, 5*(1), 46.

Jouini, K. J. J. o. I. S., & Engineering. (2022). Aggregates Selection in Replicated Document-Oriented Databases. *38*(2).

Kanade, A., Gopal, A., & Kanade, S. (2014). *A study of normalization and embedding in MongoDB.* Paper presented at the Advance Computing Conference (IACC), 2014 IEEE International.

Karnitis, G., & Arnicans, G. (2015). *Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation.* Paper presented at the Computational Intelligence, Communication Systems and Networks (CICSyN), 2015 7th International Conference on.

Kim, H.-J., Ko, E.-J., Jeon, Y.-H., & Lee, K.-H. (2018). Techniques and guidelines for effective migration from RDBMS to NoSQL. *The Journal of Supercomputing*, 1-15.

Mason, R. T. (2015). *NoSQL databases and data modeling techniques for a document-oriented NoSQL database'.* Paper presented at the Proceedings of Informing Science & IT Education Conference (InSITE).

Mehmood, N. Q., Culmone, R., & Mostarda, L. (2017). Modeling temporal aspects of sensor data for MongoDB NoSQL database. *Journal of Big Data, 4*(1), 8.

Mior, M., Salem, K., Aboulnaga, A., & Liu, R. (2017). NoSE: Schema design for NoSQL applications. *IEEE Transactions on Knowledge and Data Engineering, 29*(10), 2275-2289.

Oliveira, F., Oliveira, A., & Alturas, B. (2018). Migration of relational databases to NoSQL-methods of analysis. *Mediterranean Journal of Social Sciences, 9*(2), 227-235.

Rocha, L., Vale, F., Cirilo, E., Barbosa, D., & Mourão, F. (2015). A Framework for Migrating Relational Datasets to NoSQL1. *Procedia Computer Science, 51*, 2593-2602.

Stanescu, L., Brezovan, M., & Burdescu, D. D. (2016). *Automatic mapping of MySQL databases to NoSQL MongoDB.* Paper presented at the Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on.

Stanescu, L., Brezovan, M., & Burdescu, D. D. (2017). An Algorithm For Mapping The Relational Databases To MongoDB--A Case Study. *International Journal of Computer Science & Applications, 14*(1).

Truică, C.-O., Apostol, E.-S., Darmont, J., & Pedersen, T. B. J. B. D. R. (2021). The Forgotten Document-Oriented Database Management Systems: An Overview and Benchmark of Native XML DODBMSes in Comparison with JSON DODBMSes. *25*, 100205.

Yoon, J., Jeong, D., Kang, C.-h., & Lee, S. (2016). Forensic investigation framework for the document store NoSQL DBMS: MongoDB as a case study. *Digital Investigation, 17*, 53-65.

Younas, M. (2019). Research challenges of big data. In: Springer.