A Method for Transforming Object-relational to Document-oriented Databases

Aicha AGGOUNE
LABSTIC Laboratory,
University of 08th may 1945
PB 401, Guelma, Algeria
aggoune.aicha@univ-guelma.dz
0000-0003-2422-2591

Mohamed Sofiane NAMOUNE
Computer science departement
University of 08th may 1945
Guelma, Algeria
namoune.sofianemohamed@gmail.com

***Abstract*—Object-relational databases have emerged to improve relational ones by adding properties of object-oriented approach such as references, polymorphism, inheritance, etc. However, these extended relational databases have become a huge amount of data and the database management systems (DBMS) cannot handle them. Due to the emergence of NoSQL databases for ensuring the storage and the processing of large data scale, it is necessary to propose a method for transforming object-relational to NoSQL databases. This paper presents a new method for transforming the object-relational database to one of the popular NoSQL data stores so-called document-oriented database. The proposed method is based on a set of matching between the schemata of object-relational and document-oriented databases. The method is terminated by the generation of a set of JSON files which represent collections of semi-structured documents. These files can be imported and represented by BSON format that will be managed by document-oriented DBMS such as MongoDB.**

Keywords—Object-relational databases, Document-oriented databases, Data transformation, NoSQL data, matching.

I. INTRODUCTION

The relational databases (RDB) which based on the relational model have dominated the industry for many years. Using the relational database management system (RDBMS) allows the creating and the processing of RDB via structured query language (SQL). However, these relational databases have some limitations when querying many relational tables related between them by a join operation [1]. The most obvious limitations are [2]: (i) Simple schema which accepts only the atomic data followed the first normal form introduced by Codd, (ii) The relationships between relations are based on the join operation which complicates inserting and querying data, (iii) Querying based on declarative language which cannot use algorithm concepts such as condition (if-then-else), loop block (for, repeat, loop), function, procedure, etc.

Indeed, the querying of traditional relational tables is difficult to combine many relational tables, achieve transactions, and reduce the execution time of queries, etc. An extension of the relational model called object-relational one appears for dealing with these problems [3].

Object-relational model based on both relational model for preserving a relational structure of data and object-oriented approach for supporting its important techniques [2]. The object-relational database is a collection of object relations related between them by references and whose content can be defined by a structured data types such as Object, Reference, and a collection data types for describing the many-to-many and one-to-many associations between object-relational tables [3].We can quote the important collection data type: Nested table and Varray from Oracle DBMS.

However, the increased usage of the internet, social networking, and cloud computing involve a large volume of data with a variety of data type. The direct consequence is the so-called big data problem. The managing of huge amounts of object-relational databases becomes a difficult task, which can still be achieved with a high cost, time, and effort. In fact, the object-relational databases guarantee data consistency but they are hardly scalable and available at the same time due to the data representation, which is highly structured and normalized (using a fixed schema with respect of its constraints such as unique, primary key, foreign key, and check).

An alternative of this structured database is a non-relational data so-called NoSQL (not only SQL) owning a flexible schema known by "schema-less" with high availability, scalability, and replication of a large volume of data [4]. Due to the advantages of NoSQL stores, the organizations feel the need to migrate their existing relational databases to NoSQL databases. In this paper, we are focused in the transformation of object-relational to NoSQL databases, more especially to the document-oriented databases.

The remainder of this paper is divided into four sections. Section 2 presents the theoretical foundations of object-relational and NoSQL databases. Section 3 reviews related work in the context of data transformation into NoSQL. In section 4, we outline our method for transforming object-relational to document-oriented databases. Section 5 summaries the essential points of this work with some perspectives for future research.

II. THEORETICAL FOUNDATIONS

Relational database (RDB) provides the easiest way to store and process the data with fixed schemas. It is structured by tables called relations containing data which are represented in terms of tuples with a set of constraints [5]. With the increased usage of RDB, the handling of multiple relations has become a difficult task due to the rigid data model. RDB has been extended by supporting the primitives of the object-oriented approach [3]. The extended RDB called object-relational database (ORDB) provides a complex schema whose object relations can be contain objects with a unique object identifier (OID) which allows linking relations by references rather than by a join operation [2].

The object relation does not follow the first normal form (1NF) presented by the inventor of relational model Edgar F.Codd which admits only the atomic values [6]. Each attribute of object relation can be contained a complex data with collection types such as a nested table, dynamic array, inheritance between types, embedded collections, multimedia, etc. Thus, the object relation supports a dynamic part representing by constructors, functions and procedures [2]. The ORDB based on SQL-3 language that includes procedural language PL/SQL built by queries, method invocation, and algorithmic techniques, stored procedures, cursor, sequences, and packages [6].

ORDB represents a good solution for dealing with problems in RDB, especially the supporting a complex data with object programing. However, ORDB still requires a lot of development effort to guarantee the scalability and availability of huge volumes of data. An alternative of databases called NoSQL (Not Only SQL) was introduced and used for ensuring high availability, scalability, and replication of data [7].

NoSQL databases were introduced by Carlo Strozzi in 1998 and officially used in 2009 during an event on new non-relational databases [8]. Unlike to object-relational database, the data stored in the NoSQL are mostly unstructured or semi-structured [8]. The flexible schema of NoSQL data allows the modification of data structure during system usage whenever required. NoSQL databases have emerged as a leading new data storage technology, but some of their advantages may turn to disadvantages, the most important are [9]:

1) NoSQL databases are scalable with high availability but they do not provide the same level of data consistency as SQL databases due to the time needed to copy updates to other nodes in the cluster,

2) Schema less of NoSQL databases makes it more difficult to impose constraints on data,

3) The abundance of NoSQL system, but most of them are conceptually similar and implemental very dissimilar.

The NoSQL data can be stored according to four categories of data model [10, 11]:

- Key-Value model: the data are represented by a pair of key and value, where the key designated to uniquely identify the value, which can be of any data type (string, integer, Json file, etc.). The querying of this type of NoSQL data is the simplest and the fastest via the key. Redis and Riak are two popular NoSQL systems based on the key-value model.
- Wide Columnar-oriented model: this model is the inverse of the relational model in which the data are stored by columns instead of by rows. The columns are dynamic; they can be varied from one row to another. HBase and Cassandra are two examples of the wide columnar-oriented DBMS.
- Document-oriented model: is the most suitable type of data in the world of the internet. In general, this type of databases encapsulates key-value concept, while the key is an identifier of the document and the value is the document itself in JSON or BSON format.
- Graph-oriented model: this category is based on graph theory for modeling the complex data related by relationships. This model consists of a node storage engine and a mechanism for describing relations and properties attached to them. Neo4j is the NoSQL DBMS the most used for managing graph-oriented databases.

Consequently, most researchers are oriented toward the transformation of relational database to different types of NoSQL stores. Thus, due to the increased usage of internet and media networking sites, the document-oriented data are mostly used compared to other categories of NoSQL stores. In this paper, we are interested to transform object-relational database to document-oriented database.

III. RELATED WORK

Due to the advantages of NoSQL data for storing and processing a large volume of data, several researchers have focused on migrating and transforming the existing relational databases to NoSQL stores. According to the various types of NoSQL databases and each one stores data differently, we can classify the transformation approaches into four classes.

1) Approaches of SQL database transformation to Key-value stores: the existing works are very little and unclear. These approaches are based on the development of graphical user interface GUI aims at ensuring the transformation to different NoSQL categories includes Key-value stores [12-14]. Due to the simplest data form of key-value category, the need to transform relational or object-relational data to this type of NoSQL is rarely achieved.

2) Approaches of SQL database transformation to wide columnar-oriented stores: several researches have been interested in the relational database transformation and data migration to wide columnar-oriented databases. Hsu et al. proposed an approach based on the correlation-aware in Sqoop for transforming relational database to Hbase [15]. This work guarantees data migration, but the correlation technique does not allow the migration of the relational tables that are less correlated. Lee and Zheng proposed an approach based on linked lists and the relationships between relations [16]. This approach remains a theoretical solution and several practices should be applied for validating this proposal. Other similar works are based on the ETL (Extract, Transform and Load) process which data are extracted from RDB, transform to column-oriented data, and loaded by NoSQL DBMS [17, 18]. This approach takes a lot of cost and time due to the use of the ETL process of data mining. Recently, some toolkits have been developed to provide the migration of object data to columnar-oriented data, we can quote: ROM (Ruby Object Mapper) tool is based on the schema conversion [19], Impetus Kundera tool ensures the data migration by considering all relationships between tables are also tables [20]. These tools are presented without any detail of the migration process.

3) Approaches of SQL database transformation to document-oriented stores: There are several approaches of SQL-Document migration. Rocha et al developed a framework for auto-migration of relational database (RDB) to document oriented database (DODB) and vice versa [21]. This framework starts by the generation of DODB schema through the RDB one, then the data mapping to the new schema. The second step is achieved when querying the RDB, the framework transforms the SQL query to NoSQL one and querying the generated DODB. Thus, the result will be transformed to RDB. Stanescu et al [22] proposed an automatic approach for SQL data mapping from MySQL to DODB. This approach is based on the use of a set of corresponding rules and metadata. Other related works are based on applying a set of transformations over RDB design [23-25]. In addition, different object mapping frameworks have been developed to provide the conversion of object-oriented data to document-oriented data such as Waterline, GORM [26]. Recently, Fouad and Mohamed [27] propose a rules-based approach for transforming object-relational database to NoSQL document database under MongoDB. This approach misses some steps to complete the schema conversion such as the mapping of nested table to document data, the mapping of PL/SQL programs, etc.

4) Approaches of SQL database transformation to graph-oriented stores: The approaches of SQL-graph mappings are generally more limited when compared to the SQL-document migration. We can cite as an example, the work of Zhao et al., which is based on the transformation algorithm founded on conversion model with a set of joins between tables to generate child nodes related to their parent nodes [28]. Liao developed a data adapter system to make possible the automated transformation of multi-structured data in RDB and NoSQL systems [13]. Some frameworks have been developed, such as, DataNucleus and Eclipse JNoSQL [26].

As a result, the existing works in the literature aim at proposing an approach to transform relational databases to NoSQL databases, but they do not study the case of object-relational databases which generalized the relational ones. We present in this paper a novel method for transforming object-relational database (ORDB) to document-oriented database (DODB).

IV. METHOD FOR TRANSFORMING OBJECT-RELATIONAL TO DOCUMENT-ORIENTED DATABASES

The transformation of object-relational databases to document-oriented databases is a difficult task due to the complex data, the references between tables, and the nested data. In this paper, we present our method for ensuring this type of transformation.

Our proposed method is based on a set of steps organized into three modules: pre-processing module, processing module, and post-processing module.

The following figure summaries the principle of our proposal.

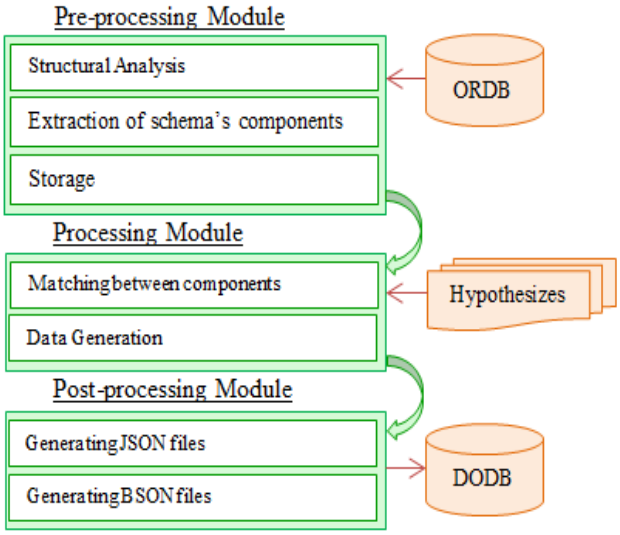

Fig. 1. General architecture of our data transformation method.

The pre-processing module aims at preparing the object-relational data to ensure the transformation by following three steps: structural analysis, extraction of schema's components and storage of results. The first step aims to analyze the structure of ORDB schema by using SQL queries. All components of object-relational schema have been determinate such as tables, Nested tables, collection types, references, attributes, etc. From this step, we can automatically detect the relationships between tables, for example, if an object relation R1 contains a nested table of R2, we can detect in the first time that R1 is a parent table. Then, the structural analysis searches the tables' names (the names of child tables) related to R2 and examine if the existing child tables contain a nested table or not. If the child tables do not contain any nested table taking the type of R1, we conclude the One-to-many association between R1 and R2 else the relation between tables is based on many-to-many association. All primary keys of object-relational tables have been extracted and they will be represented as the identifiers of the corresponding documents. Other types of constraints such as unique, check, the default value, cannot be mapped to the document-oriented data model. In order to keep these constraints, our method consists to use the existing ORDB for applying all data modification (update, insert or delete of data) and from it, we can refresh the obtained document-oriented database. As a result, all constraints of the relational model on the object-relational database have been preserved in the obtained database that is based on the document-oriented data model.

The second step allows us to extract the names of all components which are detected in the first step. In the storage step, all extracted components have been saved in lists, one of each type of the component (table names, names of attributes, Nested table names, etc.).

The processing module is a fundamental block of our method. It ensures the data transformation process according to a set of matching between ORDB and DODB components. In fact, the generation of DODB is achieved according to the following hypothesis:

1. All parent relations which contain primary key correspond to collections or JSON files. The number of files is equal to the number of these tables.

2. All tuples correspond to documents of its collection.
3. All child relations correspond to embedded document in the document which represents the parent relation.
4. All other collection types such as varray (variable array) correspond to fields with the list type.
5. Each primary key (PK) of parent table becomes the unique identifier denoted by _id, which is automatically assigned to a new document. The meaningful value of _id is the value of PK.
6. Except attributes of PK and FK (foreign key), the other attributes become fields of document.
7. Each index becomes the names of the index in the collection.
8. The foreign key is inconsiderable in the DODB model because the child table becomes a subdocument of the parent document. The embedded document takes the identifier of its principal document.
9. The NULL value which expresses the incomplete value can be express in DODB model.

After transforming the ORDB to DODB, the last module which called Post-processing consists to represent the generated DODB according to the two formats: JSON files which can be read by a simple text editor and BSON files which are handling by DBMS of DODB such as MongoDB.

To illustrate our data transformation method, we take an example of an object-relational database of business management. This ORDB contains two object relations: Enterprise and supplier. These two relations are related between them by many-to-many relationships, which is a complex association between tables compared to other associations (one-to-one, one-to-many). In this case, the object-relational schema is very complex. It is represented by the use of Nested table in the two tables (suppliers and enterprises): the suppliers table contains an additional attribute of the type Nested table of enterprise table on the one hand; the enterprise table contains an additional attribute of the type Nested table of suppliers table on the other hand.

For illustrating how our method achieves the data transformation from ORDB to DODB, we firstly need to present the object-relational schema of business management.

The data are stored in ORDB according to the following schema, which is created by using Oracle DBMS:

```
// Creating types
CREATE OR REPLACE TYPE T-Entreprise;
CREATE TYPE Ens-Entreprises AS TABLE OF REF T-Entreprise;
CREATE TYPE T-Supplier AS OBJECT (code INT, Name VARCHAR2(40), Last-name VARCHAR2(60), Phone NUMBER, Entreprise Ens-Entreprises);
CREATE TYPE Ens-Suppliers AS TABLE OF REF T-Supplier;
CREATE TYPE T-Entreprise AS OBJECT (Code INT, Title VARCHAR2(90), Suppliers Ens-Suppliers);
// Creating tables
CREATE TABLE Entreprise OF T-Entreprise
NESTED TABLE Suppliers STORE AS TheSuppliers;
CREATE TABLE Supplier OF T- Supplier
NESTED TABLE Entreprises STORE AS TheEntreprises;
```

Fig. 2. Example of Object-relational data

In the figure above, we have firstly created data types as an example T-Entreprise is an object type which contains three attributes: Code, Title and suppliers. The last attribute represents a nested table of suppliers. The creating of the table Entreprise of the T-Entreprise type by SQL-3 language under Oracle is achieved by two expressions: CREATE TABLE and NESTED TABLE. The data of the nested table store in appendix table baptized "TheSuppliers".

The data transformation method starts by the post-processing which gives a set of ArrayList of each component of the object-relational schema. In this module, we use a set of SQL queries such as SELECT tname FROM tab. This query allows extracting all existing tables in our ORDB. Then the result has been saved in the ArrayList.

In the processing module, we apply a set of matching between information in array list and document-oriented model in order to generate DODB. Whereas the post-processing module shows a generated data in JSON files which take the same name of each object relation, i.e., Entreprise.json and supplier.json.

An example of the content of Entreprise.json is illustrated in the following figure.

```
{{"_id"=976, "Title"="Samsung",
"Suppliers"=[{"Code"=36810, "Name"="John", "Last-name"="Cohen", "Phone"=34256789,
"Entreprises"=[{"code"=977, "Title"="Apple"},
{"code"=978, "Title"="IBM"}]}, {"Code"=36811,
"Name"="Mark", "Last-name"="Bron", "Phone"=67554323,
"Entreprises"=[{"code"=800, "Title"="Huawei"}]}}}
```

Fig. 3. Example of generated document-oriented data

For managing the constraints of the relational or object-relational model, all changes on the object-relational database imply the same changes in the document-oriented database. In fact, we can preserve the constraints that are lacked in the document-oriented model.

V. CONCLUSION

In this paper, we presented a novel method to transform the object-relational database (ORDB) to document-oriented database (DODB). The proposed method is mainly composed of three modules. The pre-processing module aims at preparing ORDB for the transformation to DODB. It consists of analyzing the ORDB schema and extracting all its components such as tables, attributes, relations, etc. The processing module ensures the operation of data transformation by using a set of matching based on hypotheses between components of the ORDB as well as of the document-oriented model. The post-processing module allows the representation of the transformed database in JSON and BSON formats. An example of data transformation has been presented in this paper.

In the future research, we will focus on implementing a java application validating our proposal using Oracle as an ORDBMS and MongoDB as DODBMS and trying to evaluate performance with a huge amount of object-relational databases. The second line of research which seems worth pursuing is to study the transformation of processing such as queries, transaction, PL/SQL, etc.

REFERENCES

[1] Elmasri, R., and Navathe, S., "Fundamentals of database systems". Addison-Wesley Publishing Company, 2010.

[2] Aggoune, A., "Bases de données avancées. Course", University of Guelma, Algeria, 2018.

[3] Sumathi, S., and Esakkirajan, S., "Fundamentals of Relational Database Management Systems". Springer Science & Business Media, 2007.

[4] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R.E., "Bigtable: a distributed storage system for structured data". Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Vol. 7, Seattle, 2006, pp. 15-15.

[5] Codd, E.F., "Relational database: a practical foundation for productivity: Readings in Artificial Intelligence and Databases", Elsevier, 1989, pp. 60-68.

[6] Fuh, Y.-C., Deßloch, S., Chen, W., Mattos, N.M., Tran, B.T., Lindsay, B.G., DeMichel, L., Rielau, S., and Mannhaupt, D.: "Implementation of SQL3 structured types with inheritance and value substitutability", Morgan Kaufmann Publishers Inc., 1999, pp. 565-574.

[7] Han, J., Haihong, E., Le, G., and Du, J.: "Survey on NoSQL database", 6th International Conference on Pervasive Computing and Applications, IEEE, Port Elizabeth, South Africa, 2011, pp. 363-366.

[8] Xiang, P., Hou, R., and Zhou, Z.: "Cache and consistency in NOSQL", 3rd International Conference on Computer Science and Information Technology, IEEE, Chengdu, China 2010, pp. 117-120.

[9] AGGOUNE, A., and NAMOUNE, M.S., "From Object-relational to NoSQL Databases: A Good Alternative to Deal with Large Data". Proc. Conference on Innovative Trends in Computer Science (CITCS'2019), Guelma, Algeria, 2019.

[10] AGGOUNE, A., and NAMOUNE, M.S., "Practical study for handling of NoSQL data on the distributed environment systems". Proc. 2nd edition of Conference on Informatics and Applied Mathematics, IAM' 19, Guelma, Algeria, 2019

[11] NAMOUNE, M.S., "Migration de données SQL vers des données NoSQL", Master's degree, University of 8th May 1945, 2019.

[12] Kuderu, N., and Kumari, V., "Relational Database to NoSQL Conversion by Schema Migration and Mapping", Int. J. Comput. Eng. Res. Trends, 2016, 3, pp. 506-513.

[13] Liao, Y.-T., Zhou, J., Lu, C.-H., Chen, S.-C., Hsu, C.-H., Chen, W., Jiang, M.-F., and Chung, Y.-C., "Data adapter for querying and transformation between SQL and NoSQL database", Future Generation Computer Systems, 2016, 65, pp. 111-121.

[14] Van Hieu, D., Smanchat, S., and Meesad, P.: "MapReduce join strategies for key-value storage", 11th International Joint Conference on Computer Science and Software Engineering, IEEE, Chon Buri, Thailand 2014, pp. 164-169.

[15] Hsu, J.C., Hsu, C.H., Chen, S.C., and Chung, Y.C.: "Correlation Aware Technique for SQL to NoSQL Transformation", 7th International Conference on Ubi-Media Computing and Workshops IEEE, Ulaanbaatar, Mongolia, 2014, pp. 43-46.

[16] Lee, C.-H., and Zheng, Y.-L.: "Automatic SQL-to-NoSQL schema transformation over the MySQL and HBase databases", IEEE International Conference on Consumer Electronics - Taiwan, IEEE, 2015, pp. 426-427.

[17] Li, C.: "Transforming relational database into HBase: A case study", IEEE International Conference on Software Engineering and Service Sciences IEEE, Beijing, China 2010, pp. 683-687.

[18] Mpinda, S.A.T., Maschietto, L.G., and Bungama, P.A., "From relational database to columnoriented nosql database: Migration process", International Journal of Engineering Research & Technology (IJERT), 2015, 4, pp. 399-403.

[19] https://rom-rb.org/

[20] https://github.com/Impetus/Kundera

[21] Rocha, L., Vale, F., Cirilo, E., Barbosa, D., and Mourão, F., "A framework for migrating relational datasets to NoSQL", Procedia Computer Science, 2015, 51, pp. 2593-2602.

[22] Stanescu, L., Brezovan, M., and Burdescu, D.D.: "Automatic mapping of MySQL databases to NoSQL MongoDB", Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, Gdansk, Poland 2016, pp. 837-840.

[23] Hanine, M., Bendarag, A., and Boutkhoum, O., "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, 2016, 9, 12, pp. 2369-2373.

[24] Karnitis, G., and Arnicans, G.: "Migration of relational database to document-oriented database: Structure denormalization and data transformation", IEEE, proceedings of the 7th International Conference on Computational Intelligence, Communication Systems and Networks 2015, pp. 113-118.

[25] Schram, A., and Anderson, K.M.: "MySQL to NoSQL: data modeling challenges in supporting scalability", Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity, ACM, Tucson, Arizona, USA 2012, pp. 191-202.

[26] Reniers, V., Van Landuyt, D., Rafique, A., and Joosen, W., "Object to NoSQL Database Mappers (ONDM): A systematic survey and comparison of frameworks", Information Systems, 2019

[27] Fouad, T., and Mohamed, B.: "Model Transformation From Object Relational Database to NoSQL Document Database", The 2nd International Conference on Networking, Information Systems & Security, ACM, Rabat, Morocco, 2019, pp. 49.

[28] Zhao, G., Lin, Q., Li, L., and Li, Z.: "Schema conversion model of SQL database to NoSQL", the Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE, Guangdong, China, 2014, pp. 355-362.