Indian Journal of Science and Technology, *Vol 11(21), DOI: 10.17485/ijst/2018/v11i21/121596,* June *2018*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Data Migration Tool to Minimize the Spatial Redundancy without Affecting the Query Performance

Adel A. Sewisy*, T. Ahmed, Aly S. Abdelrahim and Waleed F. Awwad

Faculty of Computers and Information, Assiut University, Egypt; Dr.adelsewisy@yahoo.com, Taloba@aun.edu.eg, aly_saber@yahoo.com, Wafs_73@yahoo.com

Abstract

Objectives: The study is to minimize the spatial redundancy without affecting the query performance, i.e., time taken to execute the query**. Methods/Statistical Analysis:** This paper illustrates a new method of analyzing the source database system: data, relations, etc., and generation of a Graph model based upon the database and conversion of the model by creating a new algorithm compatible with NoSQL databases, keeping in mind the following parameters: (1) Less time to execute each query, (2) Maximize spatial performance with minimum space and (3) Robustness of algorithm i.e., no data loss, metadata loss, or relational data loss during two transition stages, that is from old data migration to new model and analysis of new model (comparison with relational model). **Findings:** The advantage of NoSQL model is the decreased time taken to execute each query and increased speed for execution of queries due to ability to deal with the unstructured data. It was found that out of the three scenarios considered in the study that is nested, non- nested and hybrid, hybrid scenario is the one that possesses the most consistent performance and hence can be stated to be the best performer. Further, the study presented a robust algorithm for the migration of data in such a way that data loss, meta data loss, or relational data loss during transition is null or minimum. **Application/Improvements:** The present study proposes a method to convert the relational database to NoSQL database schema conversion model. This specific model has used the concept of embedded documents to improve query speed of NoSQL database.

Keywords: Data Migration, MongoDB, Relational Database, NoSQL, SQL

1. Introduction

Data Migration is the method of relocating data from one system to another wherein its database, storage or application could be changed. According to IBM Global Technology Services (2007) "Data migration is the process of making an exact copy of the current data from one device to another device-preferably without disrupting or disabling active applications—and then redirecting all input/output (I/O) activity to the new device". In simplest words, data migration is the process of transferring data from one database to another database. Data migration is usually conducted when a new system is introduced. This can include consolidation or application migration in which some older systems is replaced by a newer system. Irrespective of the nature of data migration, the aim of the migration process is to improve the performance which can be in terms of space, time or to increase the robustness.

Many studies were made on Oracle data migration[1–8]. In[1] states that now the systems have started to move from the traditional RDBMS to NoSQL wherein the movement from Oracle to NoSQL was considered. The study further stated that based on the requirements and the data structure, different NoSQL database can be used as data stores. Proper mappings should also be done while migrating from Oracle to NoSQL. In[2] created an abstraction layer between SQL and NoSQL databases. The study stated that

*Author for correspondence

an efficient and successful data migration process could help the organizations and businesses to move to the most recent systems with negligible disturbance in the work. Further, the risks related to managing the multifaceted business process can be tackled[3]. The study stated that Data in the legacy system is frozen and then extracted during this period after which the data is loaded to the target system, verification is done and then the target system is implemented. Before conducting the migration, the data should be cleaned while accumulating enough information about the data so that the control can be maintained. Also mock implementations should be conducted before the actual implementation so that the actual implementation can be shielded from the risks along with conducting proper testing like unit testing, post migration testing and functional testing[7,8].

In[4] suggested two migration strategies namely big bang migrations and trickle migrations. Big bang migrations engage implementation of the whole migration in a miniature, distinct processing window. This includes system downtime in case of system migration while the data is taken out from the source system. After this, the data is processed and loaded to the target system. This process is further followed by the transferring of processing over to the new environment. Trickle migrations take into account an incremental approach. As opposite to the big bang migration, a trickle migration includes running the old and new systems together and migrating the data in steps.

In[5] conducted a study on development of an application for data migration. This application was custom-made for a company called Dipcon to cater its data migration needs. The author stated that the maximum time required to develop a data migration tool is during the collection of information about the databases and systems. The study suggested landscape analysis to be done to avoid this problem while taking into consideration a more thorough data profiling period.

According to a study conducted on best practices in data migration[6], it was stated that for data migrations to be successful, data quality and data profiling are to be included. Further, the data migration practices should support the iterative nature since deployment of data is iterative as it includes numerous handoff and rollout phases that requires a lot of time and should involve software automation for enhancing and correcting data. Finally, the study presented Extraction, Transformation and Loading (ETL) technologies to be used for data migration because of its exceptional capability to take care of severe necessities of data migration like multi-pass data transformations, profound data profiling, terabyte-scale datasets, interoperability with data quality tools and many-to-many data incorporation abilities.

The present investigation is to minimize the spatial redundancy without affecting the query performance, i.e., time taken to execute the query.

2. Methodology and Implementation

The source database used in the present study has been created especially to conduct the migration. Microsoft SQL has been used for creating the data and storing it. The dataset thus created consists of seven tables namely countries, departments, employees, job history, jobs, locations and regions. The description of the tables has been presented in the Table 1.

3. Generation of a Graph Model based Upon the Database

The graph model created here is based on directed acyclic graph. A directed acyclic graph is a directed graph that does not contains any cycles. Each node in a graph corresponds to a Table in the database, and each edge in the graph corresponds to a Foreign Key reference in the DB. The Graph will be directed, and for each edge, source node represents the parent table and destination node represents the child table.

4. Conversion of the Model by Creating a New Algorithm to One Compatible with NoSQL Databases

The present study is based on developing an algorithm to migrate data from SQL which is a relational data base to NoSQL. NoSQL or not only SQL is a non relational database that is capable of dealing with different types of databases like ones stored in format of document, key-value, and graph and columnar formats. The algorithm thus developed has the ability to minimize the time taken to execute the query, conducting migration in such a way so as to minimize data loss, metadata loss, or rela-

Table 1. Summary table of dataset

S. No	Table Name	Attributes	Data types	Keys
1	Countries	COUNTRY_ID COUNTRY_NAME REGION_ID	CHAR(2) VARCHAR2(40) NUMBER	Primary Key- country_id Foreign Key- region_id
2	Department	DEPARTMENT_ID, DEPARTMENT_NAME MANAGER_ID LOCATION_ID	NUMBER(4) VARCHAR2(30) NUMBER(6) NUMBER(4)	Primary Key- department_id Foreign Key- location_id
3	Employee	EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, DEPARTMENT_ID	NUMBER(6) VARCHAR2(20) VARCHAR2(25) VARCHAR2(25) VARCHAR2(20) DATE VARCHAR2(10) NUMBER(8,2) NUMBER(2,2) NUMBER(6) NUMBER(4)	Primary Key- employee_id Foreign Key- department_id, job_id, manager_id
4	Job History	EMPLOYEE_ID START_DATE END_DATE JOB_ID DEPARTMENT_ID	NUMBER(6) DATE DATE VARCHAR2(10) NUMBER(4)	Primary Key- employee_id, start_date Foreign Key- job_id, employee_id, department_id
5	Jobs	JOB_ID JOB_TITLE MIN_SALARY MAX_SALARY	VARCHAR2(10) VARCHAR2(35) NUMBER(6) NUMBER(6)	Primary Key- job_id
6	Locations	LOCATION_ID, STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID	NUMBER(4) VARCHAR2(40) VARCHAR2(12) VARCHAR2(30) VARCHAR2(25) CHAR(2)	Primary Key- location_id Foreign Key- country_id
7	Region	REGION_ID, REGION_NAME	NUMBER VARCHAR2(25)	Primary Key- region_id

tional data loss during transition. In the present study, MongoDB is an open source document-oriented NoSQL database. Further, in migrating data from a traditional RDBMS to NoSQL (MongoDB in present study), the co-relations between the two have been presented as below in Figure 1. For converting relational database into NoSQL database, Author in[8] proposed a general schema conversion approach.

4.1 Minimize Time Taken to Execute each Query

One of the reasons for conducting data migration is to improve the performance. A performance of a system can be said to be improved if the time taken to extract the output is reduced. In case of MongoDB, it is known to deal with large and unstructured data wherein the traditional RDBMS lacks behind. Thus MongoDB saves time in dealing with the unstructured data where RDBMS lacks behind. Further, its ability to store large amount of data in form of documents further makes it a faster choice. MongoDB keeps as much data in RAM as possible which makes the execution of queries faster thereby saving time and improving the performance of overall migration algorithm. Also, MongoDB has the facilities of integrated storage engines and it has the ability to reduce the time between the primary failure and recovery along with in memory speed. These options further improve the

performance of MongoDB in terms of speed. The other reason is that MongoDB has lots of drivers which help in translating between programs and MongoDB which further facilitates in building the applications faster.

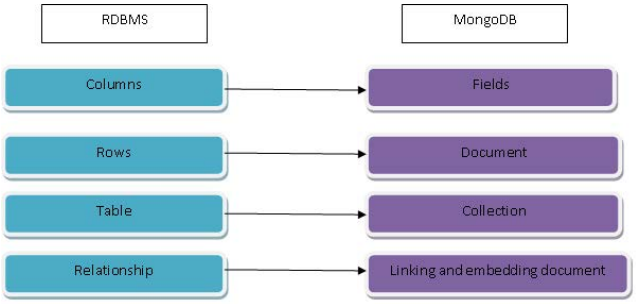

Figure 1. Correlation between traditional RDBMS and NoSQL (MongoDB).

4.2 Maximize Spatial Performance by Keeping Space Occupied to a Minimum

One of the major performance attributes is the space required for the overall process. MongoDB keeps as much data in RAM as possible. This becomes a setback for the algorithm since if the RAM space is not sufficient in the system, the overall process will slowdown. Also it takes all the RAM space which again becomes a drawback since if continuous IO requests are being generated to load the working data from the hard drive, almost all the performance in terms of speed achievements will be lost. MongoDB does this by mapping all the data into RAM and then management of the memory is done by the operating system. If, not whole dataset then at least enough RAM is required by MongoDB to hold all the database indexes. Reason being if it is required to access the hard drive just to access the index, IO performance will be blocked which will decrease the speed expectations to the minimum. Thus, irrespective of the amount of optimizations done in the code, if the data cannot be loaded fast enough from wherever it is stored, the overall performance will degrade (Figure 2).

In the algorithm above, the inputs taken into consideration are vertex and edges of the nodes along with threshold value for each weight. The threshold limit is the limit of the number of embedded tables to be executed and used at a time has been pre defined to further improve the performance of the algorithm.

Algorithm:
Input: Graph G = <V,E>, t = threshold value for edge weights
Output: Transformed Graph T=<V,E>

```
        P= { v | u,v ∈ V ∧ ~(<u,v> ∈ E) }
        C ← φ
        R ← φ
        d(u) = Set of nodes in the graph in DFPS order with u as initial node
        f(u) = { v | v ∈ V ∧ <v,u> ∈ E }
1       for v in P do
2               D ← d(v)
3               for u in D do
4                       for w in f(u) do
5                               if w = max(f(u)) ∧ w > t do
6                                       C ← w
7                               else
8                                       R ← w
9                               end if
10                      end for
11              end for
12      end for
13      T ← <V,C>
14      output T
```

Figure 2. The algorithm.

4.3 Robustness of Algorithm

The robustness of the algorithm used for the migration in the present research study is based on conducting the migration in a way that data loss, metadata loss, or relational data loss during transition is null or minimum. The robustness of the algorithm in the present study is based on the inbuilt feature of NoSQL that is dynamic schema. The property of the accepting dynamic schema by NoSQL eradicates the requirement to pre-define the arrangements like value types or fields. Such property facilitated the algorithm to inculcate hierarchical relationships representation and facility to alter or modify the records or their structure by merely adding or deleting fields. The algorithm became more robust due to the document oriented nature of MongoDB as it stores the tables or relational structures in minimal number of documents called Book. This document oriented approach helps in presenting the complex hierarchical relationship at a single place thereby eliminating the requirement of pre defining the schemas. This again reduces the intake of time and implementation of algorithm. Further, the NoSQL offers the options of auto-sharding, embedding, and on-board replication which has further facilitated in making the algorithm more robust leading to improved scalability and high availability.

5. Data Migration to New Model

The migration of the data from traditional RDBMS to NoSQL (MongoDB), has been done using the algorithm

above. In directed acyclic graph, when out degree of a vertex is zero; it is called a leaf node. Thus, a table that corresponds to a leaf node has not been referenced to any other table in the graph. In the present study, firstly, all the leaf nodes have been converted to NoSQL database according after which the edges that reference to them have been removed. The process was repeated until all the nodes are processed.

The present algorithm has been based to solve the issue of converting data from a traditional RDBMS to NoSQL database while preventing the loss of data. The output of the process will present a sequence of foreign keys, which will then present a sequence of data integration from child to parent node. It has been assumed in the study that there is no data loss among such data integration (Figure 3).

a) Completed dataset point: A point u becomes completed dataset point, if and only if u does not have any child, or every child of u as v has already become completed dataset point before data in v integrates into u.

b) Set f(u) equals to the point set of direct predecessor of u.

$$f(u) = \{ v \mid v \in V \wedge < v, u > \in E \}$$

Set F(u) equals to the point set of predecessor of u.

$$F(u) = f^*(u) = \{ v \mid v \in f(u) \vee (w \in f^*(u) \wedge < v, w > \in E) \}$$

Set b(u) equals to the point set of direct successor of u.

$$b(u) = \{ v \mid v \in V \wedge < u, v > \in E \}$$

Set B(u) equals to the point set of direct successor of u.

$$B(u) = b^*(u) = \{ v \mid v \in b(u) \vee (w \in b^*(u) \wedge < w, v > \in E) \}$$

Figure 3. The definition.

Array O[v] means the out degree of point v.

Point set Q is the set of points who's out degree is not 0. Point set P is the set of points who's out degree is 0 and in degree is not 0.

Point set T is the set of points who's out degree and in degree are 0. Sequence S is the foreign key sequence.

6. Results

This section consists of the results derived from the experiment on a sample SQL database and converted it to MongoDB, including the robustness of the algorithm, and performance with respect to query time and space as Table 2.

Table 2. SQL experiment queries

Query	No. of Rows
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME FROM EMPLOYEES WHERE JOB_ID='SA_REP';	30
SELECT E.EMPLOYEE_ID, E.FIRST_NAME, E.SALARY, J.MIN_SALARY, J.MAX_SALARY FROM EMPLOYEES E, JOBS J WHERE E.JOB_ID = J.JOB_ID AND E.MANAGER_ID='148';	6
SELECT E.EMPLOYEE_ID, E.FIRST_NAME, E.SALARY, J.JOB_ID, JH.START_DATE, JH.END_DATE, JH.DEPARTMENT_ID FROM EMPLOYEES E, JOBS J, JOB_HISTORY JH WHERE E.JOB_ID = J.JOB_ID AND E.EMPLOYEE_ID=JH.EMPLOYEE_ID;	10
SELECT E.EMPLOYEE_ID, E.FIRST_NAME, E.SALARY, J.JOB_ID, JH.START_DATE, JH.END_DATE, D.DEPARTMENT_NAME, L.CITY FROM EMPLOYEES E, JOBS J, JOB_HISTORY JH, DEPARTMENTS D, LOCATIONS L WHERE E.JOB_ID = J.JOB_ID AND E.EMPLOYEE_ID =JH.EMPLOYEE_ID AND JH.DEPARTMENT_ID=D. DEPARTMENT_ID AND D.LOCATION_ID =L.LOCATION_ID;	10

6.1 Robustness

Robustness of the algorithm can be defined as no data loss occurring as a result of the data migration. This can ascertained by the running the same SQL query on both the relation schema and the NoSQL schema and comparing the results. As shown in Table 1, 4 SQL queries were run both in relational schema and NoSQL schema, resulting in the same output in terms of both relation data and metadata.

6.2 Analysis of New Model: Comparison with Relational Model Performance

This section provides a comparative study of the performance between 3 cases, viz., Nested, Non-Nested

and Hybrid. Nested refers to a scenario where all data is embedded into their respective parent records irrespective of the weightage of their foreign key. Non-Nested refers to a scenario where all data is stored in a relational format, i.e., in separate collections in the same manner as the relational schema. Both of these approaches are not used by the algorithm. The third scenario, Hybrid refers to algorithm used in the present study, i.e., according to the weightage given to each relation, a table might either be embedded or stored in a separate collection.

The results shown in the Figure 1 implies that by the Hybrid approach, queries take a little longer to execute as compared to a Nested solution, but is definitely better than a Non-Nested solution. This decrease in query performance when compared to a Nested solution is made up for by the increase in space performance, as shown in the Tables 3 and Figures 4 and 5.

Figure 4. Query performance of nested MongoDB, Non-Nested MongoDB and Hybrid MongoDB.

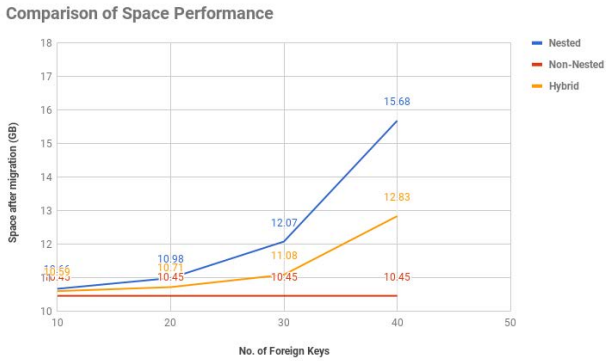

Figure 5. Space performance of nested MongoDB, non-nested MongoDB and hybrid MongoDB.

It can be inferred from the table above that though nested scenario took least query time, it took maximum space while non nested took maximum query time, it took minimum space. Thus both nested and non nested compensated for their good and bad performances in query time and space performance. The hybrid scenario is the one that possess the most consistent performance and hence can be stated to be the best performer.

Table 3. Performance comparison table

Performance		
Nested	**Non- Nested**	**Hybrid (present case)**
Least Query time (40.55 ms)	Maximum Query Time (300.57 ms)	Intermediate Query Time (148.39 ms)
Maximum space used (15.68 GB)	Minimum space used (10.45 GB)	Intermediate space used (12.83 GB)

6.3 The New Model is Better in Terms of Performance when Compared with the Relational Model

For example, considering the countries table, its information will be presented in a tabular format as follows in a traditional RDBMS. The tabular design is not easily changeable as it follows a fixed pre defined schema as Table 4.

Table 4. Traditional RDBMS structure

COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3

Following, is the format in which, MongoDB stores the data. As it can be seen that its structure can be easily changed as per the requirement as it is not rigid.

{
COUNTRY_ID: AR,
COUNTRY_NAME: Argentina,
REGION_ID: 2,
}

Above is a single document which is equal to a single row in RDBMS. A collection includes large number of such documents in the same way as a table consists of many rows. Thus, it is possible for MongoDB for one document to have supposed three fields while the other document can have five fields. The fields can be easily added, removed and modified anytime thus making it more effective and efficient when compared with traditional RDBMS.

The threshold limit set in the model further increases the speed and efficiency of the model. Further, the present system was found to be more robust since it includes no data loss during migration, increased speed for execution of queries since it saves time in dealing with the unstructured data where RDBMS lacks behind. Also, MongoDB keeps as much data in RAM as possible which makes the execution of queries faster thereby saving time and improving the performance and has lots of drivers which help in translating between programs and MongoDB which further facilitates in building the applications.

7. Conclusion

The present study proposes a method to convert the relational database to NoSQL database schema conversion model. This specific model has used the concept of embedded documents to improve query speed of NoSQL database. Further, the paper has presented a migration algorithms based on the embedded document method. The paper presented the advantage of NoSQL model that is the decreased time taken to execute each query and increased speed for execution of queries due to ability to deal with the unstructured data. It was found that out of the three scenarios considered in the study that is nested, non- nested and hybrid, the hybrid scenario is the one that possesses the most consistent performance and hence can be stated to be the best performer. Further, the study presented a robust algorithm for the migration of data in such a way that data loss, meta data loss, or relational data loss during transition is null or minimum. But the study found a drawback that is irrespective of the amount of optimizations done in the code, if the data cannot be loaded fast enough from wherever it is stored, the overall performance will degrade since MongoDB requires a lot of RAM space that often results in space issues and disruption in the hard disk space leading to the problems of bottleneck and backlogs. Thus future studies are needed to come up with a solution to the disk space related issues.

8. References

1. Cognizant. A path to efficient data migration in core banking; 2016. Available from: https://www.cognizant.com/whitepapers/a-path-to-efficient-data-migration-in-core-banking-codex2287
2. IBM Global Technology Services. Best practices for data migration; 2007. Available from: https://www-935.ibm.com/services/us/gts/pdf/softek-best-practices-data-migration
3. Mittal N, Anupindi RS, Velumani K. Oracle data migration-A comparative study; 2017 Available from: https://www.infosys.com/industries/aerospace-defense/white-papers/Documents/oracle-data-migration-comparative-study
4. Oracle. Successful Data Migration. Oracle; 2011. Available from: http://www.oracle.com/technetwork/middleware/oedq/successful-data-migration-wp-1555708
5. Rintamäki L. Data migration, a practical example from the business world master of science thesis in software engineering and technology; 2010. Available from: http://publications.lib.chalmers.se/records/fulltext/126754
6. Russom P. Best practices in data migration best practices in data migration table of contents. Available from: http://download.101com.com/pub/tdwi/files/tdwi_monograph_bpindatamigration_april20062006.
7. Roijackers J. Bridging SQL and NoSQL [Master thesis]. Eindhoven University of Technology. Department of Mathematics and Computer Science; 2012.
8. Zhao G, Lin Q, Li L, Li Z. Schema conversion model of SQL database to NoSQL. IEEE 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC); 2014. p. 355–62.