SQL to NoSQL : What to do and How

Jiao Dai
School of Cyber Security, University of Chinese Academy of Sciences, 100193, Beijing, China
daijiao0403@126.com

Abstract. Due to complex logic of enterprise management, enterprise applications may contain thousands of tables with mass connections or constraints among them. Traditional enterprises still use RDBMS while NoSQL system is widely adopted due to its excellent performance and high availability. So tools for migrating from RDBMS to NoSQL database is on demand. However, schema matters not only in RDBMS but also in NoSQL system for better query response. In this paper we discuss technology in database transformation including schema transforming, query translation and query optimization. And we mainly focus on the open issues in the context of OLAP workload and column-oriented NoSQL.

1. Introduction
With the rapid growth of data, processing and analyzing data with RDBMS (e.g., MySQL) becomes inefficient. Large internet firms like Google and Facebook create and advocate NoSQL(Not only SQL) systems such as BigTable [25] and Cassandra [26]. NoSQL system is widely adopted due to not only its excellent performance but also outstanding ability of horizontal scaling-out, high availability and fault tolerance. On the other hand, traditional enterprises are impeded by their complicated applications on its way to NoSQL. Data schema and queries in traditional enterprises are usually very intricate due to long lasting RDBMS. And schema in NoSQL actually matters considering the performance of the data access. Re-design schema for traditional enterprises is a huge challenge.

Most traditional enterprises still use RDBMS for business such as banking system. Energy industry, such as smart grid and petroleum industry, has an overwhelming urge to act due to the exploding sensor data. NoSQL database is a possible solution and enterprises need a tool to migrate data from RDBMS to NoSQL database for performance in terms of database usage.

There are many efforts on SQL on NoSQL and data migration from RDBMS to NoSQL. High-level query languages such as HiveQL [20, 21], Pig [22] and Phoenix [24] have been designed to provide friendly interfaces for developers. [10] introduces schema management on top of schema-less document systems. Apache Sqoop [27] enables users to migrate data from RDBMS to NoSQL database with the predefined schema in NoSQL. Scavuzzo et al [3] provide an approach to migrate data among different column-oriented NoSQL databases.

Database transformation, with schema design involved, obtains more and more interests in recent years. It is initially developed to cope with relational schemas [6, 9] and then extended to deal with nested XML data [1, 4]. Prototype system [7, 9] is developed based on these researches to automate the data transformation. At the same time, little work has done to transform a RDBMS into a NoSQL database. [11,12] focus on automatically transforming a relational schema to a data model supported by a NoSQL system. These works mainly focus on the three aspects: schema mapping, space occupy and performance.

In this paper, we will discuss these key points in detail. To the best of our knowledge, we are the...
first attempt to give a bird’s eye view of database migration from RDBMS to NoSQL. The rest of this paper is organized as follows. Section 2 introduces the background of NoSQL. Section 3 shows the overview on database migration. Section 4 describes related works in detail. Section 5 concludes the technologies for database transformation and its future work.

2. Preliminary
It is well known that RDBMS follows the Normal Forms while NoSQL database is denormalized. There might be inconsistent data during a short time in NoSQL. Therefore NoSQL systems are friendly to OLAP: write-once, read-many.

NoSQL database can generally be divided into four categories: key–value, document-oriented, column-oriented, and graph-oriented databases. Key-value database is suitable for storing schema-less data because data is represented as a pair of key and its value. Document-oriented database stores data in the form of document, and each document is a set of data items. So it is pretty natural to use it for web documents and logs. Column-oriented database organizes data in column family, and it is actually a sparse, distributed, multidimensional sorted map. Graph-oriented database is able to describe and store data using graph structures. Redis, MongoDB, HBase and Neo4j are separately typical databases above.

In this paper we focus on OLAP scenario rather than OLTP where data consistency is critical. And the target NoSQL database is column-oriented.

3. System Overview on Database Migration
Figure 1 gives a high level illustration of the database migration from traditional rational database system to NoSQL-based data management system. In the scenario of database migration, the schema of the tables in the DBMS, workload and data stored is the input of the system.

Given the above inputs, database migration system produces three outputs. Firstly, the schema for the underlying database is recommended. For example, in the context of column-based NoSQL system, that is picking the appropriate row-key, the column sets and their column families. Secondly, queries in SQL should be translated into operations in NoSQL. Furthermore, the query plans over big data computing platforms (e.g. MapReduce framework) will be worked out, considering the optimization over queries, the data placement policy and etc.

4. How to Migrate
The core of database transforming includes schema transforming, query translation and query optimization.

4.1. Schema Transform
Relational databases are often normalized to eliminate data redundancy, so as to save storage space and reduce the cost for update. A relation is usually decomposed into several small tables, which are
connected by foreign keys. The most simple and direct method for schema transforming is map the tables in a one-to-one manner. And make use of the primary keys and formulate new keys in NoSQL. Transform the columns in the same table into separate column families or one clustered column family.

However, it is well known that big tables, or so called wide tables, can take advantages of the underlying NoSQL storage system. And NoSQL systems do not efficiently support join operations. So the idea of denormalization, which duplicates data so that one can retrieve data from a single table without joining multiple tables, can help avoid the unnecessary cost consuming joins, is widely adopted.

There are table-level and column-level denormalization. For table-level denormalization, WideTable [28] uses aggressive denormalization to flatten a database schema into one or more big (wide) tables. Instead of generating fully materialized denormalization, A-Store [30] resorts to virtual denormalization by treating array indexes as primary keys. This design allows us to harvest the benefit of denormalization without sacrificing additional RAM space. However these works are both for in-memory data processors.

For column-level denormalization, it is the problem of deciding what column families to create, and what information to store in each column family, for a given application. In general, this will depend on how the application needs to use the data. CLDA [29] avoids join operations without denormalizing entire tables by duplicating only columns that are accessed in non-primary-foreign-key-join predicates. CLDA also combines tables that are modified within the same transaction into a unit of atomic updates to support atomicity.

4.2. Query Translation
As we mentioned before, HiveQL, Phoenix and etc. translate SQL-like queries into MapReduce tasks or operations in HBase. Although these works have been done to make SQL on NoSQL easier, complex SQL queries are still open problems. And query rewrite is very important for further optimization.

TPC-H, as the de facto benchmark for database transformation from RDBMS to NoSQL consists of queries with subqueries and views. As for subqueries, query unnesting techniques [31, 32] works, and temporary table is for views. WideTable[28] tackles scans, group-by operations and aggregations, leaving other queries to the archival part rather than the WideTable Baking component.

[12] proposes JackHare framework with SQL query compiler, JDBC driver and a systematical method using MapReduce framework for processing the unstructured data in HBase.

4.3. Query Optimization
Query optimization is critical in view of performance. In general, query rewrite for multiple queries, data compression for less redundancy and data placement are commonly used.

AQUA [20] is a query optimizer which aims at reducing the cost of storing intermediate results by optimizing the order of join operations. QMapper [19] is an intelligent translator to enable automatic rule based SQL-to-HiveQL mapping as well as cost-based optimizing of the translated queries.

Although denormalization might simplify query processing by converting join operations into scan operations, it can slow down the access to each individual tuple due to much larger tuple in the denormalized table. WideTable uses a columnar storage representation with dictionary encoding to control the space overhead.

CA_Sqoop [5] analyze the log to know which tables are frequently used for JOIN, which is a frequently used operation which merges two tables according to specific columns. Then it distribute above tables on the same node if possible to enhance data locality

5. Conclusion
Database migration is a hot topic in big data processing. With the continuous development of data storage and management technology, it attracts the attention of industry enterprises. Migrating applications from traditional RDBMS to big data platform raises a dilemma: obeying normalization may loss the benefit of big data platform, thus violates the original intention; inappropriate denormalization may bring about vast cost of data manipulation, sometimes may even lead to a
disaster. In this paper, we figure out the data migration system framework, application scenarios, and an in-depth analysis of different aspects of the problem. We summarize advantages and disadvantages of existing solutions and applicable scenario. Optimization among multiple queries will be important in future work.

6. References

[1] M. Arenas and L. Libkin, XML data exchange: consistency and query answering. In: 28th ACM Symposium on Principles, pp. 13--24 (2005)
[2] L. Cabibbo, On keys, foreign keys and nullable attributes in relational mapping systems. In: 12th International Conference on Extending Database Technology, pp. 263--274 (2009)
[3] M. Scavuzzo, E. Di Nitto, S. Ceri, Interoperable Data Migration between NoSQL Columnar Databases. In: 18th IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations. pp.154--162. IEEE Press, New York (2014)
[4] A. Fuxman, M. Hernandez, C. Ho, R. Miller, P.Papotti and L. Popa, Nested mappings: schema mapping reloaded. In: 32nd International Conference on Very Large DataBases, pp. 67--78 (2006)
[5] M J. C. Hsu, C. H. Hsu, S. C. Chen and Y. C. Chung, : Correlation Aware Technique for SQL to NoSQL Transformation. In: 7th International Conference on Ubi-Media Computing and Workshops(UMEDIA), pp. 43--46. IEEE Press, Ulaanbaatar (2014)
[6] Kolaitis P G : Schema mappings, data exchange and metadata management. In: 24th ACM Symposium on Sigmod-sigact-sigar on Principles of Database Systems, pp. 61--75 (2005)
[7] R. Miller, M.Hernandez, L.Haas, L. Yan, C. Ho, R. Fagin, L. Popa,: The Clio project: managing heterogeneity. ACM SIGMOD Record, 2001, vol. 30, pp. 78—83 (2001)
[8] Paola Gómez, Rubby Casallas, Claudia Roncancio,: Data schema does matter, even in NoSQL systems!.In: 10th IEEE International Conference on Research Challenges in Information Science, Grenoble, France (2016)
[9] L. Popa, Y. Velegrakis, R. Miller, M.Hernandez, R. Fagin,: Translating web data. In 28th International Conference on Very Large DataBases, pp. 598-609 (2002)
[10] L. Wang, S. Zhang, J. Shi, L. Jiao, O. Hassanzadeh, J. Zou, and C. Wangz,: Schema management for document stores. In the VLDB Endowment, vol. 8, no. 9, pp. 922—933, (2015)
[11] G. Zhao, Q. Lin, L. Li, and Z. Li,: Schema conversion model of sql database to nosql. In 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 355--362 (2014)
[12] W.C. Chung, H.P. Lin, S.C. Chen, M.F. Jiang, and Y.C. Chung,: Jackhare: a framework for sql to nosql translation using mapreduce. Automated Software Engineering, vol. 21, no. 4, pp. 489–508 (2014)
[13] Kim H-J, Ko E-J, Jeon Y-H, Lee K-H,: Migration from RDBMS to column-oriented NoSQL: lessons learned and open problems. EDB, LNee, vol 461, pp 25–33 (2017)
[14] Ho-Jun KimEun-Jeong KoYoung-Ho JeonKi-Hoon Lee,: Techniques and guidelines for effective migration from RDBMS to NoSQL. In The Journal of Supercomputing. Springer US.(2018)
[15] Zhao G, Lin Q, Li L, Li Z,: Schema conversion model of SQL database to NoSQL. In: 9th International Conference on P2P, IEEE , pp 355--362 (2014)
[16] Zhao G, Li L, Li Z, Lin Q,: Multiple nested schema of HBase for migration from SQL. In: 9th International Conference on P2P. IEEE , pp 338–343(2014)
[17] Mior MJ, Salem K, Aboulnaga A, Liu R,: NoSE: schema design for NoSQL applications. In: 32nd IEEE Transactions on Knowledge & Data Engineering, pp 181—192 (2016)
[18] Lee C-H, Zheng Y-L,: Automatic SQL-to-NoSQL schema transformation over the MySQL and HBase databases. In: 2nd IEEE International Conference on Consumer Electronics-taiwan, pp 426--427 (2015)
[19] Yingzhong Xu, Songlin Hu,: QMapper: A Tool for SQL Optimization on Hive Using Query Rewriting. In: 28th International Conference on World Wide Web Companion. pp 211--212 (2013)
[20] Wu S, Li F, Mehrotra S, et al.: Query optimization for massively parallel data processing. In:
2nd ACM Symposium on Cloud Computing. ACM, pp 12 (2011)

[21] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., et al.: Hive - A Petabyte Scale Data Warehouse Using Hadoop. In: 26th IEEE Transactions on Knowledge & Data Engineering, pp 996--1005 (2010)

[22] Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A not-so-foreign Language for Data Processing. In 35th ACM Conference on Management of Data, pp 1099--1110 (2008)

[23] Lee, K.H., Kim, S.Y., Whang, E., Lee, J.G.: A Practitioner’s Approach to Normalizing XQuery Expressions. In: 11th International Conference on Database Systems for Advanced Applications. LNCS, vol. 3882, pp. 437--453 (2006)

[24] Phoenix, https://phoenix.apache.org/

[25] Chang F., Dean J., Ghemawat S., et al. BigTable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems, 26(2), pp. 1-26(2008)

[26] Cassandra, https://cassandra.apache.org/

[27] Sqoop, https://sqoop.apache.org/

[28] Li Y. WideTable: An Accelerator for Analytical Data Processing. Proceedings of the Vldb Endowment, pp 907--918 (2015)

[29] Yoo, J., Lee, K.-H., Jeon, Y.-H.: Migration from RDBMS to NoSQL using column-level denormalization and atomic aggregates. Journal of Information Science and Engineering. 34(1), pp. 243--259 (2018).

[30] Zhang Y., Zhou X., Zhang Y., et al. Virtual Denormalization via Array Index Reference for Main Memory OLAP. IEEE Transactions on Knowledge & Data Engineering, 28(4): 1061--1074(2016)

[31] Lee, K. H., Kim, S. Y., Whang, E., & Lee, J. G.: A Practitioner’s Approach to Normalizing XQuery Expressions. Database Systems for Advanced Applications. Springer Berlin Heidelberg(2006)