Automatic SQL-to-NoSQL Schema Transformation over the MySQL and HBase Databases

Chao-Hsien Lee, *Member, IEEE*, and Yu-Lin Zheng

***Abstract*—The explosive growth of huge data lets cloud computing be more and more popular in recent years. Traditional web-based content management systems (CMS), e.g., phpBB, WordPress, and Joomla, store data using relational databases whose key advantage is the strong relationships among tables. However, regarding the flexibility and the feasibility of parallel processing, cloud computing adopts NoSQL databases that can support horizontal scaling to handle big data. Therefore, how to transform the existing SQL data into the NoSQL database becomes an emerging and necessary issue. This paper is motivated to propose an automatic SQL-to-NoSQL schema transformation mechanism over the MySQL and HBase databases. Based on our experimental results, the proposed mechanism is able to improve about 47% access performance.**

I. INTRODUCTION

In order to reduce the redundancy among data, traditional relational databases organize data into more than one table, which is composed of rows and columns. In each table, every row represents one instance and is identified with a unique key, i.e., primary key. Columns in a row represents the attribute values of each instance. Database normalization is the explicit process to find out the relationships among tables and these relationships are also known as foreign keys. Once the database normalization is finished, i.e., data are organized into tables, relational databases can be accessed using structured query language (SQL). That is, users can utilize the JOIN operation to query data across tables simultaneously. Hence, relational databases are also called SQL databases.

Not Only SQL (NoSQL) databases provide different mechanisms rather than the tabular relationships of relational database for the data storage. Generally speaking, based on the data model, NoSQL databases can be divided into four categories, i.e., (1) column, (2) key-value, (3) document and (4) graph, in which a column-oriented NoSQL database is also made up of tables. In other words, the column-oriented NoSQL is most similar to the traditional relational database. Regarding the explosive growth of huge data, e.g., the information collected from the Internet of Things (IoT), NoSQL supports horizontal scaling, i.e., processing several instances on different servers simultaneously. In order to satisfy the aforementioned auto-scaling characteristic, the column-oriented NoSQL database stores data into a big table instead of several tables in the relational database. Thus, no cross-table query is required in the column-oriented NoSQL database. In contrast to the database normalization of relational databases, as far as we know that the column-oriented NoSQL database only has the design principles of (1) Denormalization, (2) Duplication and (3) Intelligent Keys (DDI). No standardized or explicit procedure is able to produce the column-oriented NoSQL's table schema.

Regarding the main difference between the SQL and column-oriented NoSQL schemas, this paper is motivated to propose one automatic SQL-to-NoSQL schema transformation mechanism. In other words, the proposed mechanism is able to generate the column-oriented NoSQL schema automatically based on the imported SQL schemas. This paper is organized as follows. Section II shows related works about SQL and NoSQL. Section III describes our automatic SQL-to-NoSQL schema transformation in details. Section IV presents the performance evaluation. Finally, Section V concludes this paper.

II. RELATED WORKS

Recently, due to the emergency requirement of big data, more and more researchers focus on NoSQL. For example, Lombardo et al. [1] discussed issues about complex data structures in the NoSQL database. Li and Manoharan [2] analyzed the performance between SQL and key-value based NoSQL databases. Furthermore, Naheman and Wei [3] considered the comparison between HBase and other NoSQL databases, e.g., BigTable, Cassandra, CouchDB, MongoDB, etc. Based on their observation and analysis, NoSQL databases may not always get better performance than SQL databases. Therefore, Hsu et al. [4] proposed and designed the correlation-aware technique for Sqoop which is utilized to transform the data stored in the traditional relational database into the HBase. The proposed correlation-aware technique is able to analyze which tables are frequently used for the cross-table query and then transform these data into the same Hadoop data-node. On the other hand, Gadkari et al. proposed a theoretical model to implement the cross-table query over the HBase and get the produced results faster.

III. AUTOMATIC SQL-TO-NOSQL SCHEMA TRANSFORMATION

In order to avoid the cross-table query in the NoSQL database, our design is motivated to follow the NoSQL's DDI design principles, i.e., to aggregate related tables into a big table and then automatically select the most suitable key, which is called row key, to identify each row. This paper takes HBase, which is one of the most popular column-oriented

C.-H. Lee and Y.-L. Zheng are with Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan. Correspondence: chlee@mail.ntut.edu.tw (Prof. C.-H. Lee)

The research is supported by the Ministry of Science and Technology of the Republic of China (Taiwan) under the grant numbers MOST 103-2218-E-027-009.

978-1-4799-8745-0/15/$31.00©2015 IEEE

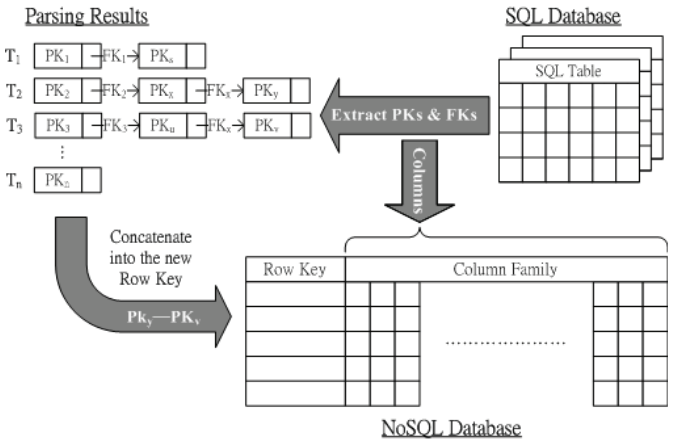

Fig. 1. The automatic SQL-to-NoSQL schema transformation.

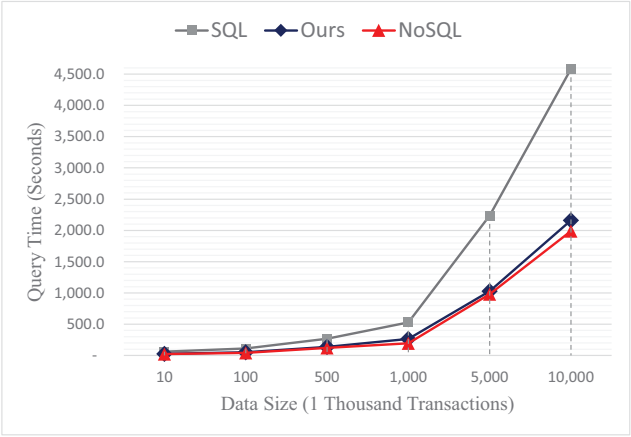

Fig. 2. The query performance comparison over the Hush database.

NoSQL, into consideration. Regarding how to design the table schema, two basic choices are (1) tall-narrow and (2) flat-wide. The tall-narrow design pattern means a table with few columns but many rows. On the other hand, the flat-wide design pattern has fewer rows but many columns. Since HBase can only split at row boundaries, our proposed mechanism should enforce to transform the imported table schemas from the SQL database into the tall-narrow design pattern no matter which design pattern is utilized in the SQL database.

In order to achieve the above goal, the selected row key must preserve the highest cardinality. Figure 1 depicts the proposed automatic SQL-to-NoSQL schema transformation mechanism. First, the proposed mechanism parses the SQL table schemas automatically and converts the relationships among tables into several linked lists. In our implementation, MySQL stores all table schemas in one special table called "information_schema". Thus, we can extract each table's primary key (pk) and foreign key (fk) and then let each table's primary and foreign key become the head of one linked list. Next, we check the foreign key and continuously concatenate the related tables' primary and foreign keys into the corresponding linked list. After parsing all tables, we are able to get the chained length of all linked lists. According to our design, the row key with the highest cardinality should be the combination of all primary keys with the longest chained length. For example, as depicted in Figure 1, PK_y in the T_2 linked list has the longest chained length and PK_v in the T_3 linked list also has the same longest chained length. Therefore, the automatically-selected row key is PK_y-PK_v in this example.

IV. Performance Evaluation

Since one database called Hush from the HBase textbook [6] provides two-version schemas, i.e., the SQL version and the NoSQL version, at the same time, this paper implements our proposed mechanism over the Hush database on the fair concern. First, for the SQL version, we directly convert the Hush database schema into the HBase and utilize the HiveQL to perform the SELECT and JOIN operations, i.e., the cross-table data query. Second, for the NoSQL version, we implement the Hush database schema in the HBase. Due to the DDI design principles of NoSQL, no cross-table data query is required in this experiment. Finally, we transform the SQL version into the HBase based on our proposed mechanism.

Figure 2 depicts the query performance comparison over the Hush database. The data size is configured from 10 thousand transactions to 10 million transactions. When the data size increases, three mechanisms require more time to complete the data query. However, the SQL version requires much more time than other two mechanism while the data size is 10 million transactions. As described in Section II, our proposed mechanism is devised according to the design principles of NoSQL. Thus, the proposed mechanism has almost the same performance as the NoSQL version and improve about 47% performance related to the SQL version.

V. Conclusion

Although most web network services are still based on the SQL database, NoSQL provides better scalability and performance to handle big data. In order to reduce the migration overhead from SQL to NoSQL, this paper is motivated to provide an automatic SQL-to-NoSQL schema transformation mechanism. Based on our experimental results, the proposed mechanism can determine one suitable row key and then improve 47% access performance.

Reference

[1] S. Lombardo, E. Di Nitto, and D. Ardagna, "Issues in Handling Complex Data Structures with NoSQL Databases," *Proceedings of the 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 443-448, Sept. 2012

[2] Yishan Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 15-19, Aug. 2013.

[3] W. Naheman and Jianxin Wei, "Review of NoSQL databases and performance testing on HBase," *Proceedings of International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, pp. 2304-2309, Dec. 2013.

[4] Jen-Chun Hsu, Ching-Hsien Hsu, Shih-Chang Chen, and Yeh-Ching Chung, "Correlation Aware Technique for SQL to NoSQL Transformation," *Proceedings of the 7th International Conference on Ubi-Media Computing and Workshops (UMEDIA)*, pp. 43-46, Jul. 2014.

[5] A. Gadkari, V.B. Nikam, and B.B. Meshram, "Implementing Joins over HBase on Cloud Platform," *Proceedings of IEEE International Conference on Computer and Information Technology (CIT)*, pp. 547-554, Sept. 2014

[6] Lars George, *HBase: The Definitive Guide*, O'Reilly Media, 2011.