SQL-to-NoSQL Schema Denormalization and Migration: A Study on Content Management Systems

Chao-Hsien Lee and Yu-Lin Zheng
Department of Electronic Engineering
National Taipei University of Technology
Taipei, Taiwan, R.O.C.
Correspondence: chlee@ntut.edu.tw

***Abstract*—Content management systems (CMSs) are able to let people, who have no technical skill to manage websites, rapidly create, edit, organize and publish online contents. For example, CMSs are often used to establish e-commerce online shops, community portals, personal blogs, or organization websites. Thus, CMSs play the role of collecting data and then can be easily extended to become intelligent Internet systems. Most popular CMSs, e.g., WordPress and Joomla, rely on relational databases or SQL databases, e.g., MySQL, PostgreSQL, etc. However, due to the explosive growth of huge data, the well-structured characteristic of SQL databases may limit the scalability to handle horizontal scaling. Therefore, regarding the flexibility and the feasibility of parallel processing, cloud computing takes Not Only SQL (NoSQL) databases into consideration. How to migrate the original CMS software from SQL databases to NoSQL databases becomes one emerging and critical research issue. This paper is motivated to propose an autonomous SQL-to-NoSQL schema denormalization and migration. At the same time, this paper also evaluates the proposed mechanism on the realistic CMS software. Based on our experimental results, our mechanism not only helps the current CMS software migrate into the cloud platform without re-designing their database schemas, but also improves at least 45% access performance after schema migration.**

Keywords-Schema Migration; Denormalization; SQL; Not Only SQL (NoSQL); Content Management System (CMS)

I. INTRODUCTION

With the widespread cloud services, e.g., Facebook, Google, and Amazon, web-oriented architecture (WOA) based software accelerates the deployment of huge contents. The continuous evolution of smart devices and wearable devices shifts toward the implementation of web-based mobile apps, which are developed using HTML5, CSS and Javascript, instead of native mobile apps. In order to avoid establishing a separate website for mobile apps, responsive web design (RWD) provides a framework to adapt web contents to all kinds of devices, including traditional desktops or brand-new handheld devices. A content management system (CMS) is one kind of Internet system composed of tools to help create, edit, organize and publish online contents even if people have no technical skill to manage websites. Since most CMSs allow developers to customize or add plug-ins, CMSs can be easily extended to bsystems that are responsible for information retrieval, mining and analysis. The most popular CMSs, including WordPress and Joomla, store data using relational databases, which are also called SQL databases because they support the structured query language (SQL). Data stored in a SQL database would be organized into more than one table. A table is composed of rows and columns. Each row is one instance and identified with a unique key called primary key. Columns inside a row represent the attribute values of an instance. In order to reduce the redundancy among data and minimize the storage space, normalization is the explicit process to find out the relationship among each table. In other words, after the normalization process, rows in a table can be linked to rows in other tables by

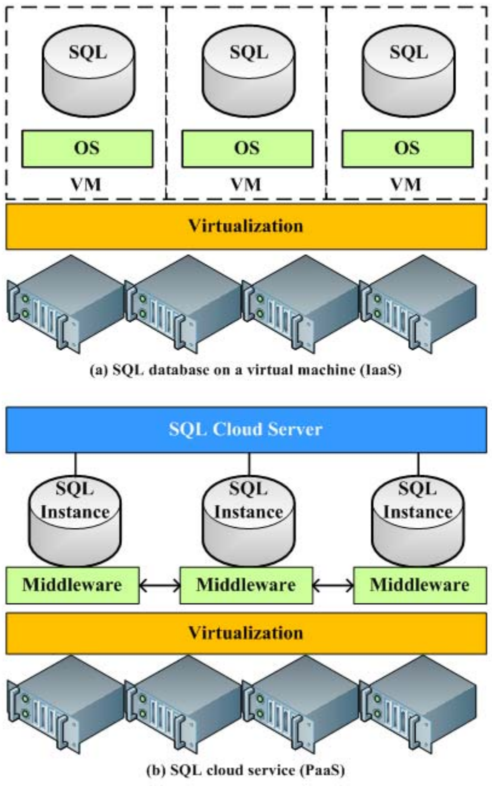

Figure 1. SQL database hosting on different cloud service models

a unique key called foreign key. The relational databases allow users to access data using the SQL, in which the JOIN operation is able to query data across tables simultaneously.

As the popularity of big data, many website administrators migrate their websites from physical machines into the cloud environment. Figure 1 depicts the SQL database hosting on different cloud service models. First, referring to Figure 1(a), the SQL database service can be hosted on a virtual machine (VM). This kind of hosting is established over the cloud service model called infrastructure as a service (IaaS) and suitable for the fast migration to the cloud environment with minimal changes. For example, Microsoft Azure enables subscribers to deploy their SQL servers over Azure VMs. On the other hand, as depicted in Figure 1(b), the SQL database service can be provided directly by the cloud platform. This kind of hosting is established over the cloud service model called platform as a service (PaaS) and able to reduce overall costs for managing several SQL databases. For example, Microsoft Azure also provides Azure SQL Database. Subscribers can store their data directly into Azure SQL Database without deploying any SQL server. The goal of migrating the SQL database service to the cloud environment is the flexibility while scaling up or out for greater computing power. However, no matter which kind of hosting is adopted, the SQL database should still keep the ACID properties, i.e., (i) atomicity, (ii) consistency, (iii) isolation and (iv) durability, which guarantee each database transaction is processed reliably.

Not only SQL (NoSQL) databases provide different mechanisms rather than the tabular relationships of SQL databases. Regarding the data model, NoSQL databases can generally be divided into (1) key-value, (2) document, (3) graph, and (4) column-oriented. First, in the key-value NoSQL database, e.g., CouchDB, data can be represented by (i) a string called key and (ii) the actual data content called value. In other words, the key-value NoSQL database is suitable for storing schema-less data. Secondly, the document NoSQL database, e.g., MongoDB, aggregates data into semi-structures, i.e., combining data and schema into a document. Third, the graph NoSQL database, e.g., Neo4j, is able to describe and store data using graph structures. Finally, similar to the SQL database, the column-oriented NoSQL database is also composed of tables. However, the number of columns may change row by row. That is, the concept of a table in the column-oriented NoSQL database is vague and flexible. Instead of the ACID requirements, NoSQL databases typically follow another alternative BASE model, i.e., (i) basic availability, (ii) soft-state and (iii) eventual consistency. Since the BASE model releases the requirement of consistency after every transaction, NoSQL databases are able to support horizontal scaling, i.e., processing several instances on different servers simultaneously.

Based on the aforementioned comparison between SQL and NoSQL databases, since intelligent algorithms usually require more computing power to shorten the reaction time, migrating the current CMSs to the cloud platform and enjoying the advantages of parallel processing become one emergent research issue. Thus, this paper would focus on the column-oriented NoSQL database and is motivated to propose an autonomous SQL-to-NoSQL schema denormalization and migration. The realistic SQL database schemas of WordPress and Joomla are taken into consideration in this paper. The proposed mechanism can read the database schemas from MySQL, and then automatically analyze and produce the schemas for HBase. At the same time, we evaluate the access performance with different size of data and compare them with and without using the JOIN operation of Hive that provides an SQL-like language called HiveQL. Therefore, our experimental results show that the proposed mechanism can improve the data access performance and provide the good foundation to design and develop intelligent Internet systems.

This paper is organized as follows. Section II shows related works. Section III describes our proposed SQL-to-NoSQL schema denormalization and migration in details. Section IV presents the performance evaluation. Finally, Section V concludes this paper.

II. Related Works

Due to the popularity of the NoSQL databases, a lot of works discussed and analyzed the SQL and NoSQL databases. 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 NoSQL databases. Boicea et al. [3] compared Oracle, i.e., the traditional SQL database, and MongoDB, i.e., the document NoSQL database from theoretical differences, feature restrictions and system architecture. Furthermore, Grolinger et al. [4] identified issues and challenges in handling big data using MapReduce. Four main categories are (1) data storage, (2) big data analytics, (3) online processing, and (4) security and privacy. Naheman and Wei [5] 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. Scavuzzo et al. [6] provided an approach to migrate data among different column-oriented NoSQL databases. One intermediate metamodel was proposed to represent data in a common format and responsible for the translation from a source database to the target one.

Regarding the emergent requirement of big data, more and more recent researchers focus on how to integrate SQL and NoSQL. For example, Hsu et al. [7] 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. For HBase, i.e., column-oriented NoSQL database, Zhao et al. [8] combined all related tables into different column families. Thus, the relationships in all SQL tables can be preserved completely into one single NoSQL table. Regarding SQL table nesting conditions, Zhao et al. [9] proposed a graph-based transforming algorithm. For MongoDB, i.e., document NoSQL database, the proposed algorithm may cost more space to improve the query performance after schema conversion. Sellami et al. [10] developed open PaaS database API (ODBAPI) that is a streamlined and REST-based API to be able to execute the CRUD operations, i.e., create, read, update and delete, on the SQL and NoSQL databases. Li et al. [11] designed a query-oriented data modeling (QODM) approach, in which the data stored structure and the data query requirements should be verified and then the NoSQL database schema would be produced by the proposed QODM.

In addition to migrate the SQL database into the NoSQL database, some works enabled the cross-table query over the NoSQL database. Gadkari et al. [12] proposed a theoretical model to implement the cross-table query over the HBase and get the produced results faster. Wei et al. [13] implemented a middleware layer called Cloud TPS to enable join queries and keep strong transactional consistency over NoSQL databases, e.g., SimpleDB and HBase. Regarding the document NoSQL database, Lawrence [14] presented a virtualization system which allows to query and join data using a SQL query and automatically translate into the underlying API to access MongoDB. Finally, Hieu et al. [15] considered the key-value NoSQL database and analyzed the MapReduce join strategies, including theta-join, all pair partition join, repartition join, broadcasting join, semi join, presplit semi join.

III. SQL-TO-NOSQL SCHEMA DENORMALIZATION AND MIGRATION

Referring to Section I, the SQL database has the well-structured data and supports the cross-table query. However, the well-structured characteristic of the SQL database also limits horizontal scaling. Thus, the key contribution of this paper is to denormalize SQL database schemas and then migrate into a NoSQL database autonomously.

A. Denormalization and Row-key Selection

As far as we know that there is no explicit or standard procedure for column-oriented NoSQL database to create the column-oriented NoSQL schema. In order words, the column-oriented NoSQL database only has the DDI design principle, i.e., (1) denormalization, (2) duplication and (3) intelligent keys. First, denormalization and duplication mean to re-aggregate related SQL tables into a big NoSQL table even if some data should be duplicate or redundant in the NoSQL table. Then, each row should select one intelligent key called row key for

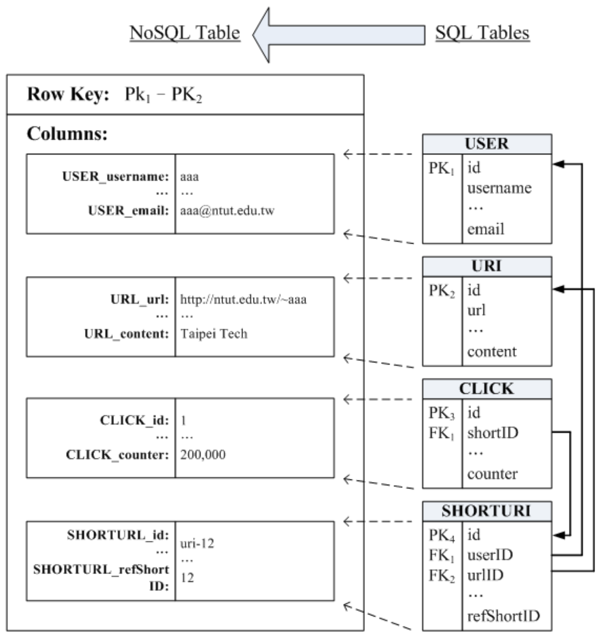

Figure 2. Schema denormalization based on the DDI design principle.

unique identification. Since all related SQL tables are merged into a big table, no cross-table query is required in the NoSQL database. Therefore, we are motivated to follow the DDI design principle and avoid any cross-table query in the NoSQL database.

Regarding the traditional table schema design, the tall-narrow design pattern lets a table have few columns but many rows. On the other hand, the flat-wide design pattern lets a table have few rows but many columns. Since HBase, which is utilized in this paper for implementation, can only split at the row boundary, the proposed mechanism will try to create the NoSQL database with the tall-narrow design pattern. In order to achieve the tall-narrow design, the selected row key must keep the highest cardinality. Since primary keys are utilized to identify each row in the SQL tables, the row key in the NoSQL table should be the concatenation of several primary keys in our proposed mechanism. Next, we take the relationship among SQL tables into consideration. For example, one SQL table called Table A is related to another table called Table B by a foreign key. However, Table B is also related to another table called Table C. Based on our observation and studies, the chained relationship is longer, the primary key of the corresponding table, i.e., Table C in this example, keeps higher cardinality. Thus, in this example, Table C's primary key would be selected as the row key in the NoSQL table. If more than one SQL table with the same chained relationship, the row key would be the concatenation of these SQL tables' primary keys.

Figure 2 depicts an example of the proposed schema denormalization based on the DDI design principle. In this example, one SQL database called Hush from the HBase textbook [16] is composed of four SQL tables, including (1) USER, (2) URI, (3) CLICK and (4) SHORTURI. The CLICK table is related to the SHORTURI table, and the SHORTURI table is related to the USER and URI tables. Thus, based on the aforementioned description, the selected row key is the concatenation of the USER and URI tables' primary keys, i.e., PK_1-PK_2 in Figure 2. At the same time, since four tables are related, all columns in four tables are aggregated into one NoSQL table.

B. Schema Migration

As depicted in Figure 3, the proposed schema migration is autonomous, i.e., create the NoSQL schema automatically after the SQL schema analysis. In our implementation, MySQL stores all SQL schemas in one special table called “information_schema”. Hence, we can parse and extract each table's primary key and foreign keys from the SQL database.

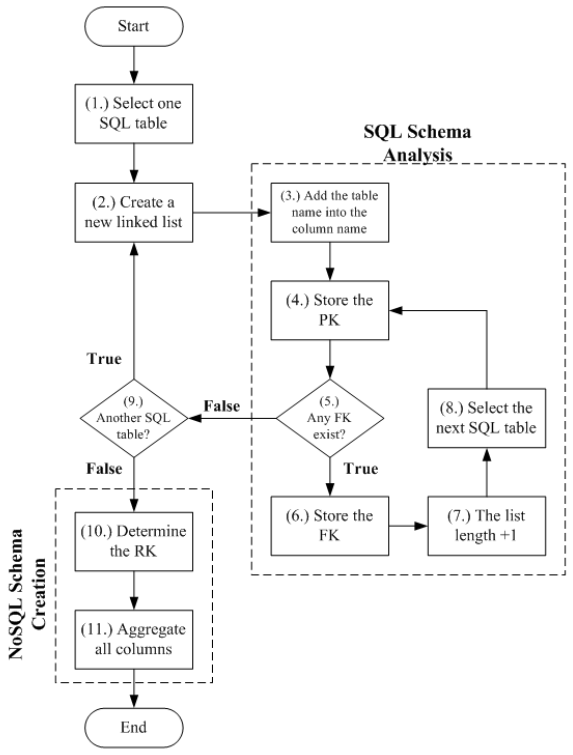

Figure 3. The flowchart of the proposed schema migration

The executed steps of our schema migration are as follows.

1) One table is selected from the SQL database for the SQL schema analysis.
2) A linked list is created to store the result of the SQL schema analysis. Referring to Section III.A, the main functionality of a linked list is utilized to calculate the chained relationship.
3) Since the same column name may be existed in different tables, the proposed migration should add the table name into each column name for clear identification.
4) The primary key (PK) is stored as the row key (RK) candidate of the NoSQL table.

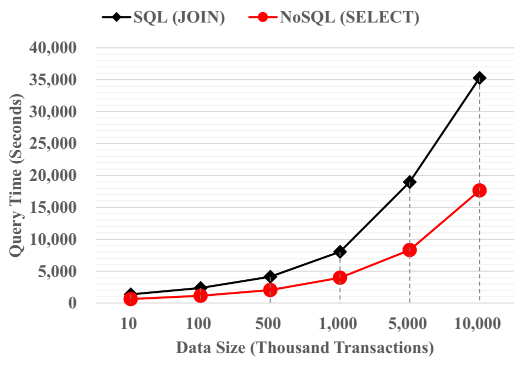

Figure 4. The performance comparison over the Joomla CMS.

5) The proposed migration determines whether there is any foreign key (FK) existed in this SQL table or not.
   a) If true, go to Step 6.
   b) If false, go to Step 9.
6) The foreign key (FK) is stored to aggregate the related table.
7) Based on the foreign key (FK), the related table is added into the linked list. Then, the list length is incremented by one.
8) We select the related table as the next table and repeat the SQL schema analysis again, i.e., go to Step 4, until all related table are added into the linked list.
9) The proposed migration determines whether there is any other table which is not analyzed yet or not.
   a) If true, go to Step 2.
   b) If false, go to Step 10.
10) Referring to Section III.A, the proposed migration determines the row key (RK) based on the stored primary keys (PKs).
11) Referring to Section III.A, the proposed migration aggregates all columns into one NoSQL table.

IV. PERFORMANCE EVALUATION

This paper evaluates the proposed autonomous SQL-to-

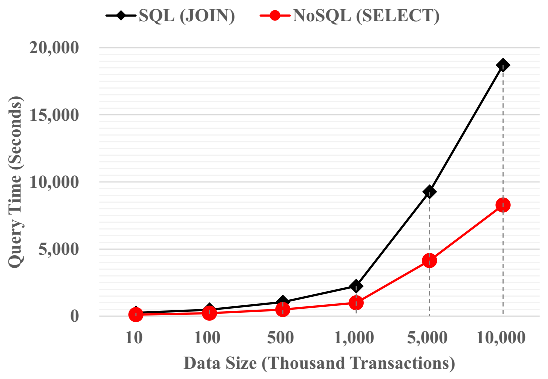

Figure 5. The performance comparison over the WordPress CMS.

NoSQL schema denormalization and migration using two popular CMSs, i.e., WordPress and Joomla. First, we migrate the SQL database schemas of WordPress and Joomla directly to the NoSQL database by the Sqoop and then utilize HiveQL to perform the cross-table query. On the other hand, we also migrate the SQL database schemas using the proposed mechanism. Since no cross-table query is required, we utilize HiveQL to perform the single-table query in the following experiments. For each CMS, we utilize the Talend Open Studio Big Data tool to randomly generate up to 10 million transactions.

A. WordPress

WordPress is widely deployed to establish websites. According to the 2015 CMS investigation, WordPress was used by more than 23% of top websites. WordPress's SQL database is composed of total 11 SQL tables. After our SQL schema analysis, WordPress's SQL tables can be divided into two groups. One group is composed of 3 SQL tables and the other one is composed of 6 SQL tables. The remaining 2 SQL tables are independent, i.e., no foreign key exists. Figure 4 depicts the performance comparison over the WordPress CMS. When the data size increases, two mechanisms require more time to complete the query. However, our proposed mechanism requires much less time while the data size is 10 million transactions. The proposed mechanism is able to save 45% query time on average.

B. Joomla

Joomla is another CMS to publish web contents and estimated to be the second most popular CMS after WordPress. Joomla's SQL database has much more SQL tables than WordPress. There are totally 34 SQL tables. Based on the results of our SQL schema analysis, Joomla's SQL tables can be categorized into three groups. The largest group is composed of 17 SQL tables. Thus, Joomla's SQL database has complex relationship. Figure 5 depicts the performance comparison over the Joomla CMS. Similar to the WordPress's results, two mechanisms require more time to complete the query when the data size increases. Regarding the Joomla CMS, our proposed mechanism is able to save 48% query time on average.

V. SCHEMA DENORMALIZATION AND MIGRATION

Content management systems (CMS) are popular Internet system to publish web contents and able to extend new functionalities, e.g., intelligent data analysis or decision support. However, most CMSs are still based on the traditional SQL databases. Since the NoSQL databases provide better scalability and support parallel processing to handle big data simultaneously. This paper is motivated to propose an autonomous SQL-to-NoSQL schema denormalization and migration. In other words, the CMS administrators or developers need not re-design their database schemas. According to the experimental results on WordPress and Joomla, the proposed mechanism is able to save 45~48% query time on average. The future work will consider how to divide columns into different column families.

ACKNOWLEDGMENT

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

REFERENCES

[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.] A. Boicea, F. Radulescu, and L.I. Agapin, "MongoDB vs Oracle -- Database Comparison," Proceedings of The 3rd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), pp.330-335, Sept. 2012.

[4.] K. Grolinger, M. Hayes, W.A. Higashino, A. L'Heureux, D.S. Allison, and M.A.M. Capretz, "Challenges for MapReduce in Big Data," Proceedings of IEEE World Congress on Services (SERVICES), pp.182-189, Jun. 2014.

[5.] 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.

[6.] M. Scavuzzo, E. Di Nitto, and S. Ceri, "Interoperable Data Migration between NoSQL Columnar Databases," Proceedings of IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), pp.154-162, Sept. 2014.

[7.] 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.

[8.] Gansen Zhao, Libo Li, Zijing Li, and Qiaoying Lin, "Multiple Nested Schema of HBase for Migration from SQL," Proceedings of 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp.338-343, Nov. 2014.

[9.] Gansen Zhao, Qiaoying Lin, Libo Li, and Zijing Li, "Schema Conversion Model of SQL Database to NoSQL," Proceedings of 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp.355-362, Nov. 2014.

[10.] R. Sellami, S. Bhiri, and B. Defude, "ODBAPI: A Unified REST API for Relational and NoSQL Data Stores," Proceedings of IEEE International Congress on Big Data (BigData Congress), pp.653-660, Jun. 2014.

[11.] Xiang Li, Zhiyi Ma, and Hongjie Chen, "QODM: A query-oriented data modeling approach for NoSQL databases," Proceedings of IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), pp.338-345, Sept. 2014.

[12.] 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.

[13.] Zhou Wei, G. Pierre, and Chi-Hung Chi, "Scalable Join Queries in Cloud Data Stores," Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp.547-555, May 2012.

[14.] R. Lawrence, "Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB," Proceedings of International Conference on Computational Science and Computational Intelligence (CSCI), pp.285-290, Mar. 2014.

[15.] D. Van Hieu, S. Smanchat, and P. Meesad, "MapReduce join strategies for key-value storage," Proceedings of the 11th International Joint Conference on Computer Science and Software Engineering (JCSSE), pp.164-169, May 2014.

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