SQL to NoSQL Transformation System using Data Adapter and Analytics

Ganesh B. Solanke
Department of Computer Engineering
Pimpri Chinchwad College of Engineering
Pune, India
solankeganesh777@gmail.com

Sudarshan S. Deshmukh
Department of Computer Engineering
Pimpri Chinchwad College of Engineering
Pune, India
Deshmukh.sudarshan@gmail.com

Abstract*— Popularity of big data in cloud computing is growing tremendously nowadays. Data production rate has become explosive. Many existing systems need to expand their service to support explosive data generation rate. NoSQL database are suitable to handle large volume of data. This paper proposes a data adapter which supports hybrid database systems, NoSQL databases and relational databases. Also, a seamless mechanism is provided for hybrid database system construction, which enables access to both databases at same time. This paper focuses on the automated transformation of multi-structured data hybrid systems. Hybrid database systems provide access to either relational database or NoSQL database, per data size. The proposed data adapter supports application queries and database transformation at same time, which in turn speed up the process. Three different approaches are proposed for queries on database. Blocking transformation mode, blocking truncation mode, direct access mode. This paper also discusses the integration of Pentaho business analytics tool with proposed data adapter, for big data predictive analytics. This integration focuses on empowering users to prepare, model and visualize and explore data sets stored in NoSQL databases. This paper describes the design of the data adapter in detail.

***Keywords*— NoSQL, Big data, Data adapter, Hybrid database, Data analytics.**

I. INTRODUCTION (*HEADING 1*)

Big data is a widespread nowadays. Big data represents the large and fast growing size of information that is mostly unused by existing analytical applications and data warehousing systems. Organizations are interested in storing, monitoring and analyzing this data because it can add significant value to the decision-making process. Such processing, however, may involve complex loads that thrust the limits of what is possible using old-style data warehousing and data management techniques and technologies.

NoSQL databases developed because of the exponential growing of the Internet and the growth of web applications. NoSQL databases provide efficient and faster access on big data. Large organizations and enterprises are moving towards NoSQL database for analysis and quicker access on big data. Cassandra is key value oriented type of NoSQL database discussed in this paper, which is highly scalable and high performance distributed database.

Instead of substituting existing relational database, enterprises and big organizations are moving towards hybrid SQL-NoSQL database solution. Hybrid SQL-NoSQL database solutions combine the advantage of being well-suited with many SQL applications and enabling the scalability of NoSQL. In hybrid solution, applications communicate with relational database to handle requests for small and scale of data. Queries demanding large amount of data are handled using NoSQL data. NoSQL database works as pool of data and provides batch operations, back-ups, and analysis.

Hybrid database designs affect the existing system design. In existing system, application use SQL queries to communicate with relational database. NoSQL database can't be accessed by SQL, and modification of the design to access both relational and NoSQL database. Data transformation mechanism is needed after integration of NoSQL with existing relational database system. The transformation suspends and blocks application for data synchronization. Large data transformation takes larger time, which in turns becomes serious issue for some real-time, unstoppable services.

Hybrid database designs affect the current system design. In existing system, application use SQL queries to communicate with relational database. NoSQL database can't be accessed by SQL, and needs to change the design to access both relational and NoSQL database. Data transformation mechanism is needed after integration of NoSQL with existing relational system. The transformation suspends and blocks application for data synchronization. Large data transformation takes larger time, which in turns becomes critical issue for some real-time, non-stopping services.

Data adapter consist following features:

1. Single SQL interface:

Single SQL interface is used to access both relational and NoSQL database. It includes SQL query parser and query translator. The Simba's Technologies Cassandra ODBC Driver [1] enables direct SQL query translation to the

Cassandra Query Language (CQL), and offers users unparalleled performance at scale. The in-built Cooperative Query Execution (CQE) passes down filters and aggregations to provide upgraded access performance to Cassandra. Existing application design does not need modification to work with NoSQL database, by using existing SQL interface.

2. Database converter:

Datastax enterprise 2.0 [2] include support for Apache sqoop [3], which is a tool designed to transfer data between an RDBMS and Hadoop. DataStax Enterprise combines Cassandra, Hadoop together into one big data platform, which can now move data to Hadoop and Cassandra as well. Apache sqoop is a tool for transferring bulk data between relational database and NoSQL database. Data synchronization is handled by synchronization mechanism after each relational database table transformation. The queries arrived during transformation are blocked and patched later.

3. Query approach:

Three different approaches are proposed for query execution. Transformation blocking approach, dump blocking approach and direct access approach.

This paper discusses all tools and approaches in detail for transformation from relational to NoSQL database and application query executions. The paper is organized as follows. Section II discusses related work. Section III presents design of proposed hybrid system using data adapter. Section IV discusses various query approaches for synchronization mechanism. Section V discusses theoretical analysis of time and overhead. Section VI presents the conclusion and future work.

II. Related Work

OpenStack [4] is open source cloud computing software and allow users to build their cloud platform. Users can design and develop big data applications in short time using OpenStack platforms. Instead of forming clusters, users can focus more on design and development of applications using online platforms of cloud computing. Hashem Yaqoob [5], has discussed comparison and analysis of multiple big data cloud platforms like Google, Microsoft, Amazon and Cloudera. Various big data tools are emerged for development of big data analytics system.CLP Chen [6], has discussed challenges, techniques and technologies for big data with principles for designing big data.

There has been huge work done on different NoSQL databases [7] like BigTable [8], HBase [9, 10], MongoDB [11], and Cassandra [12] for big data [13, 14]. NoSQL is a new breed of database that has risen to the big data challenge—the Not Only SQL (NoSQL) database. NoSQL provide efficient storage and access for big data. In this paper, Cassandra [15] is a NoSQL database in the data adapter system. Apache Cassandra database is the right NoSQL database choice for scalability and high availability without compromising performance.

NoSQL is capable to manage big data, but relational database is efficient to manage middle or small amount of data. Multiples works on hybrid database system has discussed integration of databases. R Cattell [16] has discussed scalable SQL and NoSQL data stores for simple OLTP-like applications. M Fazio [17] has discussed the need of hybrid storage approach for IoT and also proposed two-layer architecture based on hybrid storage system, which supports Platform as a Service. Architecture hybrid database system and data transformation approaches depend on the types of applications and their services. K A Doshi [18] has discussed types of applications. Doshi has classified data growth types as horizontal, vertical and chronological. Various suitable approaches are also proposed for blending SQL and NewSQL platforms depending on data growth. Synchronization is handled by integrator.

The architecture proposed in this paper integrates relational and NoSQL database, and manages continuously growing data and handle queries in real time as well. To handle queries on hybrid database system in real time, schema mapping is to be considered with migrated data. Migration of tables can be done in two ways. Either all tables in relational database can be migrated to NoSQL database or create a table in NoSQL for each table in relational database. Either to choose a new data model this can increase database workload significantly. Other way is to have an external process sync data from relational to Cassandra while running both new and old logic in parallel. A strategy for schema mapping is also proposed to translate data model from relational to Cassandra. J Pith [19] has discussed a mechanism to access NoSQL databases using proposed SQL command subset. Cassandra uses CQL (Cassandra Query Language) which is like SQL. Complex queries in relational databases can be performed using CQL. J Roijackers [20] has discussed bridging of SQL and NoSQL. NoSQL details are hidden in the work. Simba technologies' [1] has derived Cassandra ODBC and JDBC Drivers with SQL Connector. It offers direct, universal ODBC 3.8 and JDBC 4.0, 4.1 data access to Cassandra data stores through Cassandra's native API, with no extraction. Simple SQL commands can be used instead of CQL queries.

This paper focuses on the transformation where relational and NoSQL database have same copies of tables. The transformation of data is in single way from relational database to NoSQL database. This paper proposes a data adapter which uses database converter using Apache sqoop [3]. Apache soop tool is designed for efficiently transferring bulk data between NoSQL database and structured datastores such as relational databases. Many studies [21, 22] have designed hybrid database system using sqoop as data converter. Table synchronization is problem raised in database transformation. J Cho [23] has discussed database synchronization which improves freshness. Author has also discussed to refreshing a local copy of data and maintaining consistency. Authors have also discussed synchronization policies and study. This paper also focuses on addition of analytical capabilities on proposed hybrid database system. Pentaho [2] is a platform which can be integrated to visualize and analytics. Ying-Ti Liao [24] has also designed and discussed a data adapter for querying and transformation between SQL and NoSQL database. Author has provided three different modes of query approach in their proposed design.

III. PROPOSED DESIGN

In existing system application communicates only with relational database using SQL queries. Applications and data growth leads to use of NoSQL databases. Big data and complexity in transformation between databases in hybrid database systems resulted into this data adapter system.

The data adapter consists different modules and layers. It performs queries on database and data transformation between databases simultaneously. Single SQL interface is provided to communicate between both relational and NoSQL database. The components in proposed system are: relational database, NoSQL database, data adapter, application, and analytical tool. Application and database coordinates with each other in the system. Database converter performs the data transformation and reports to data adapter system. Data adapter acts as intermediate between the application and database. It parses incoming query, submit query to database and fetch results back to application. Data adapter continuously monitors the transformation progress and takes decision about query execution. Data adapter and database converter can perform simultaneously. Data adapter controls the converter. Synchronization is performed by converter to maintain the consistency.

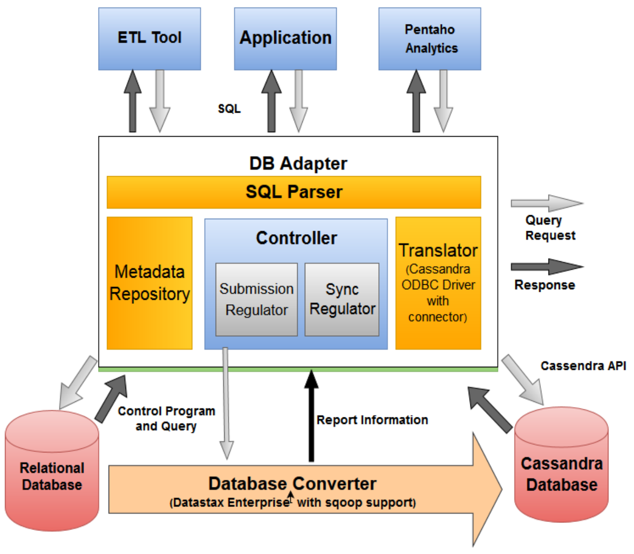

Fig. 1. Architecture and components of proposed design

Data adapter and database converter are two main parts. Database converter converts data between relational database and NoSQL database. Data adapter communicates with applications and databases. We describe design of adapter and its components each.

Apache Cassandra is a free and open-source distributed database management system. It is intended to handle huge amounts of data across many commodity servers, and provides high availability without single point of failure. Cassandra is hybrid between a key-value and a column-oriented database management system. Cassandra cannot do joins or subqueries.

Cassandra ODBC and JDBC Drivers with SQL Connector: Simba's Cassandra ODBC and JDBC Drivers present direct, universal ODBC 3.8 and JDBC 4.0, 4.1 data access to Cassandra data stores via Cassandra's native API, with no pulling out. It enables direct query translation from SQL to the Cassandra Query Language (CQL), providing users unparalleled performance at scale. The integral Collaborative Query Execution (CQE) gives down filters and aggregations to provide improved performance access to Cassandra. It is Powerful SQL Connector allows users to define schemas on the fly on schema-less data.

Database adapter communicate with relational database and NoSQL database. MySQL JDBC driver is used to connect with MySQL. Simba's Cassandra JDBC and ODBC driver is used to connect with Cassandra database. SQL queries are performed through translator, and SQL translator handles the translation of SQL queries.

SQL Parser is responsible for accepting queries from applications and parse, extract and pass to controller. Parsers understand the read and write queries. It communicates with controller to convey information regarding synchronization.

Controller is responsible for table transformation between relational database and NoSQL database. Flow of queries and table synchronization is also performed by controller. Table synchronization is performed as per query approaches. Queries related with modification, and insertion of data is added into queue. It consists two components: submission regulator and sync regulator. Submission regulator captures transformation progress in local metadata repository. After each table is transformed, Sync regulator performs synchronization process.

IV. QUERY APPROACHES FOR SYNCHRONIZATION

Relational and NoSQL database both can be accessed by data adapter, even when data transformation is being performed. Various query approaches are given due to data transformation. Data and tables can be modified unexpectedly in various stages of transformation. Data inconsistency between both databases can occur, if query performing and data transformation occurs on same time.

Following figure 2 shows how inconsistency may arise. We have relational database table on left hand side and NoSQL database on right hand side. Table is partitioned into3 parts P1, P2, P3. Database converter carry out transformation of P1, P2, P3 respectively. At time T1, transformation of P1 is done. At time T2, transformation of P2 is done. And finally, P3 is transformed. In between write query arrives which affects P2 and P3. Then the data between both database tables become inconsistent.

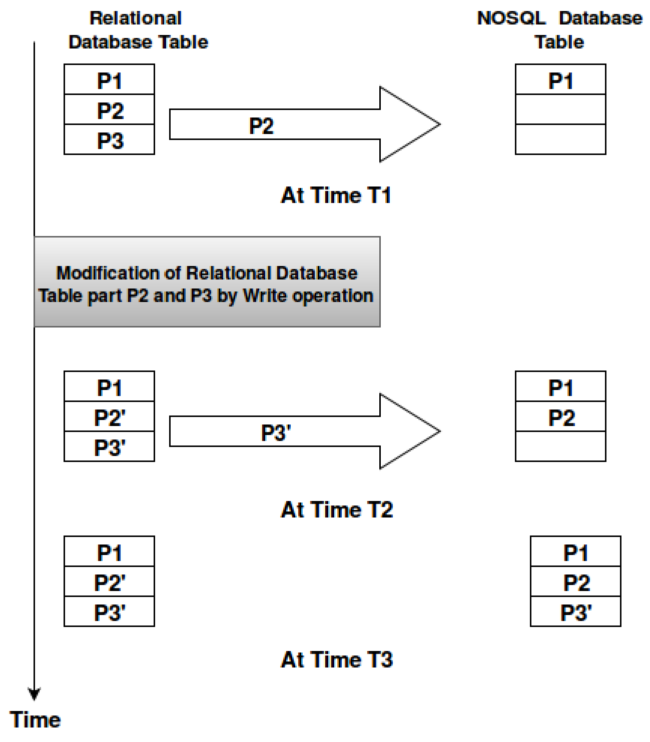
Relational Database Table
NOSQL Database Table
P1
P2
P3
P2
P1
At Time T1
Modification of Relational Database Table part P2 and P3 by Write operation
P1
P2'
P3'
P3'
P1
P2
At Time T2
P1
P2'
P3'
P1
P2
P3'
At Time T3
Time

Fig. 2. Data inconsistency problem in data synchronization

Table synchronization is needed to affect query on both database at same time. Three approaches are proposed for query. Inconsistent data can be synchronized after transformation in some approaches. Data adapter performs patching of those queries later.

1. *Transformation blocking approach:*

Requests for read operations are executed immediately. All other requests are blocked, which can affect the tables which are being transformed. There are three stages as shown in figure 3. In Waiting stage, tables wait in relational database, and not transformed. Requests are executed only on relational database. In transformation stage, table is transformed to NoSQL database. In between all queries are blocked. Table is completed transformation in finish stage. Blocked queries are performed later NoSQL database. Large time is taken by transformation in case of large data.

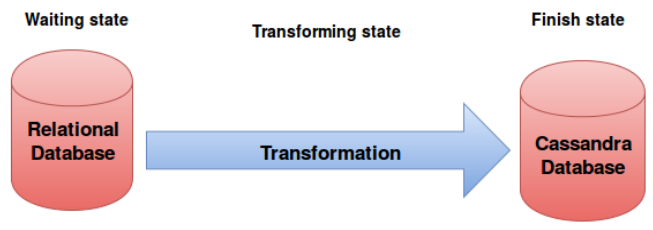

Fig. 3. Transformation Blocking Approach

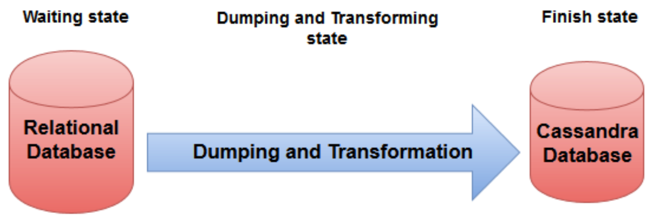

Fig. 4. Dump Blocking Approach

2. Dump blocking approach:

Transformation is separated into dump and transform. The time required for transformation is reduced here. Fig 4 shows in detail. Data are dumped from relational database to dump files in dumping stage. In transform stage, dumped data is transformed to NoSQL database. Here, patching is performed at beginning of finish stage. Queries are blocked only in dump mode, but in transform stage. But still requests from application are blocked in dump mode.

3. *Direct access approach:*

Query execution and database transformation are separated here. Requests are allowed at all time at any stage. At any stage, requests are immediately executed. Serious data inconsistency problem can be arisen. After synchronization, database will be eventually consistent. It uses synchronization mechanism to deal with this data inconsistency issue. It provides better accessibility than other approaches.

V. Theoretical Analysis

In this section, we will discuss performance in terms of synchronization time and synchronization overhead. Synchronization time is initial time when same data is in relational and NoSQL database as well. Synchronization overhead refers the number of patches.

Synchronization time:

Consider n tables. All tables are converted one by one from 1, 2……, to n. Requests are patched later after tables conversion. At next time, conversion of next table and queries on previous converted table can take place simultaneously. Patching time is less than the table conversion time. So, patching time is overlapped with next conversions. The synchronization time is total of conversion time of all tables and patching time of last table.

$$Ttotal = \sum_{k=1}^{n} tk + \propto * Pq$$

Here, tk refers kth table conversion time, refers average patching time for single query and Pq refers queries to be patched.

Pq = qq * fq

Here, qq refers number of all queries to be performed when conversion of table tn.

Optimal synchronization time is to find the table with less queries to be patched at last.

Synchronization overhead:

Count of patches is referred as overhead of synchronization. The last table synchronization overhead time can be reduced by identifying time duration t_{hf} that queries accessing T_n with maximum frequency f_n and avoid this table conversion during t_{hf}. An algorithm is proposed to optimal table order to minimize patches.

```
Step 1: For (p=1; p<=n; p++)
            For (q=i; q<=n; q++)
    Find and store all table order combinations with starting order by $T_p$
Step 2: For (p=1; p<=n; p++)
    Find maximum query frequency $f_p$ of $T_p$ in time $T_{hf}$
    Eliminate combination related to $T_p$ with maximum query frequency $f_p$ from combinations stored
Step 3: Till all combinations are not finished, Compute count of queries to be patched
Step 4: Choose table combination to be patched with minimum count of queries
```

VI. Conclusion and Future Work

In this paper, a multi-module data adapter design for hybrid database system is proposed. Without modifying existing application design, we can have a hybrid database system. Three approaches for data synchronization that: transformation blocking approach, dump blocking approach and direct access approach are also discussed. Each approach has its own different policies for blocking queries. We also discussed theoretical analysis of synchronization time as well as synchronization overhead. The algorithm is also presented for synchronization overhead minimization.

In the future, we will focus on implementation of this proposed design and evaluate with models. We will focus more on speeding up of operations. We will also focus on provision of all complicated SQL queries. Enhancements in query parser, database converter and query approaches will also be focused. We will also offer security components for preventing data from being hacked and compromised.

References

[1] “Cassandra ODBC and JDBC drivers with SQL connector”, Available: http://www.simba.com/drivers/cassandra-odbc-jdbc/

[2] “Datastax enterprise 2.0”, Available: http://www.datastax.com/2012/03/how-to-move-data-from-relational-databases-to-datastax-enterprise-cassandra-using-sqoop

[3] “Apache Sqoop”, Available: http://sqoop.apache.org/

[4] OpenStack Available: https://www.openstack.org/

[5] I.A.T. Hashem, I. Yaqoob, S. Mokhtar, N.B. Anuar, A. Gani, S.U. Khan, “The rise of ‘big data’ on cloud computing: Review and open research issues”, Information Systems 47 (2015) 98–115.

[6] C.L. Philip Chen, C. Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data”, Inform. Sci. 275 (2014) 314–347.

[7] J. Han, E. Haihong, J. Du, G. Le, “Survey on NoSQL database”, 6th International Conference on Pervasive Computing and Applications, ICPCA, 2011, pp. 363–366.

[8] F. Chang, S. Ghemawat, J. Dean, W.C. Hsieh, D.A. Wallach, M. Burrows, et al., “Bigtable: A distributed storage system for structured data”, ACM Transaction Computational Systems 26 (2008).

[9] “Apache HBase”, Available: http://hbase.apache.org/

[10] M. Vora, “Hadoop-HBase for large-scale data”, International Conference on Computer Science and Network Technology”, 2011, pp. 601–605.

[11] Kristina Chodorow, “MongoDB: The Definitive Guide”, O’Reilly Media, Inc., 2013.

[12] Avinash Lakshman, Prashant Malik, “Cassandra: a decentralized structured storage system”, ACM SIGOPS Operations System Rev. 44 (2010) 35–40.

[13] Neal Leavitt, “Will NoSQL databases live up to their promise?” Computer archival 43, Issue 2(February 2010) pp. 12–14.

[14] James Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh A. H. Byers, “Big data: The next frontier for innovation, competition, and productivity”, 2011. Mckinseyglobal institute.

[15] Apache Cassandra. Available http://cassandra.apache.org/

[16] Rick Cattell, “Scalable SQL and NoSQL data stores”, ACM SIGMOD Rec. 39 (2011), pp 12–27.

[17] M. Fazio, A. Celesti, M. Villari, A. Puliafito, “The need of a hybrid storage approach for IoT in PaaS cloud federation”, 28th International Conference on Advanced Information Networking and Applications Workshops, WAINA, 2014, pp. 779–784.

[18] K.A. Doshi, T. Zhong, Z. Lu, X. Tang, T. Lou, G. Deng, “Blending SQL and NewSQL approaches: Reference architectures for enterprise big data challenges”, 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC, 2013, pp. 163–170.

[19] J. Rith, P.S. Lehmayr, K. Meyer-Wegener, “Speaking in tongues: SQL access to NoSQL systems”, In Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC’14, 2014, pp. 855–857.

[20] J. Roijackers, “Bridging SQL and NoSQL (Master’s thesis)”, Eindhoven University of Technology, 2012.

[21] O.V. Joldzic, D.R. Vukovic, “The impact of cluster characteristics on HiveQL query optimization”, 21st Telecommunications Forum, TELFOR, 2013, pp. 837–840.

[22] T. Kim, H. Chung, W. Choi, J. Choi, J. Kim, “Cost-based join processing scheme in a hybrid RDB and hive system”.

[23] J. Cho, H. Garcia-Molina, “Synchronizing a database to improve freshness”, ACM SIGMOD, 2000, pp. 117–128.

[24] Y Lio, J. Zhou, C. Lu, S. Chen, C. Hsu, W. Chen, M. Jiang, Y. Chung, “Data adapter for querying and transformation between SQL and NoSQL database”, Elsevier, Future Generation Com. Sys., 65, 2016, pp. 111-121.