Hadoop as a Service: Integration of a Company's Heterogeneous Data to a Remote Hadoop Infrastructure

Yordan Kalmukov, Milko Marinov
Department of Computer Systems and Technologies, University of Ruse, Ruse, Bulgaria

***Abstract*—Data analysis is very important for the development of any business today. It helps to identify organizational bottlenecks, optimize business processes, foresee customers' demands and behavior, and provides summarized data that could help reducing costs and increase profits. Having this information when designing new products or services highly increases the chances of their success, and thus provides an additional competitive advantage over other businesses. However, having a single data analyst with a computer is far from enough in the era of big data. There are powerful data analytical software tools, but they are either expensive or hard to deploy and require multiple high-performance servers to run. Buying expensive hardware and software, and hiring high-qualified IT experts, is not affordable for all companies, especially for smaller ones and start-ups. Therefore, this article proposes an architecture for integration of a company's heterogeneous data (stored within a database of any type, or in the file system) to a remote Hadoop cluster, providing powerful data analytical services on demand. This is an affordable and cost-effective cloud-based solution, suitable for a company of any size. Businesses are not required to by any hardware or software, but use the data analytical services on demand, paying a small processing fee per request or by subscription.**

***Keywords*—Hadoop integration; data analytical tools; heterogeneous data integration; Hadoop distributed file system (HDFS); HBase; hive**

I. Introduction

Data acquisition and analysis are very important for the development of any business today. They help to better understand the underlying business processes and allow their subsequent optimization and reorganization. The analysis of the collected data provides the company the ability to identify dependencies among processes, objects and subjects; to optimize resource planning; to analyze consumers' needs and behavior; to foreseen customers' actions and demand; and many others. All these analyses give a significant competitive advantage to the company over the other competitors, since it can easily detect trends or bottlenecks and identify ineffective processes to optimize.

For all this to happen, however, the company must have the necessary computer hardware, software and the human resources to do the analysis. In case of processing large volumes of data, having a computer and an expert in the field is far from enough. Manual analysis of big data by a single person is practically impossible. Instead, automated, computer-based tools should be applied in order to analyze big data efficiently and effectively. However, these tools are quite expensive and not every company can afford to buy them, especially start-ups. There are free tools either, some of them very powerful, but not that user-friendly, requiring highly-qualified programmers and IT administrators to deploy and configure them.

A set of powerful data analytical tools is provided by the Hadoop ecosystem [1],[2]. They are built on top of the Hadoop Distributed File System (HDFS). Most of them are distributed as open source applications developed and maintained by community of enthusiast professionals, under the auspices of the Apache Software Foundation. The tools are quite powerful, but as distributed computing applications they should be installed and configured on multiple servers. So, a company that wishes to build its own data analytics infrastructure should buy multiple high-performance servers and hire IT experts to deploy and configure the entire cluster.

Although there are third party installation and management wizards, for example Cloudera Manager, installation of Hadoop services is far from trivial and easy. There are many dependencies that should be considered in advance. For example, since all services run on top of HDFS, if the HDFS itself is not properly installed and configured, the installation of the data analytics tools may fail. The biggest problem is that in case of failure, a subsequent reinstallation is not an option at all, since there are many files left in the server's filesystem from the previous installation. A subsequent reinstallation will just make things messier. Installation and deployment should be done by really qualified and experienced IT experts. Their wages however could be even higher than the money spent for dozens of high-performance servers. As a result, installation and maintenance of own Hadoop infrastructure could be quite expensive and not affordable for small and medium size enterprises, especially for star-up companies.

Fortunately, there is a cost-effective, more flexible and easier way to use Hadoop ecosystem's data analytics tools than building a company's own Hadoop infrastructure. The cloud computing model can help a lot here. An experienced IT company, a service provider, could build the entire Hadoop infrastructure on its own servers, and then offers all the Hadoop's data analytics tools to other companies as services for rental. The idea could be named "Hadoop as a Service (HaaS)".

How could an external organization/company take advantage of the data analytics resources of the service provider? By integrating its data to the Hadoop cluster of the service provider. And this could be done at much lower cost for the external company than if it had to build its own infrastructure for data analysis. During integration however, it should be taken into account that both the Hadoop infrastructure and the company's database are already existing systems. Not just existing systems, but heterogeneous systems. The latter is very important and implies that some intermediate layer should be developed and deployed on top of the Hadoop cluster that will allow external applications/users to communicate with it regardless of their own architecture and implementation technologies. Furthermore, there is another key requirement as well – together with the data integration, the service provider should support data separation. Data separation means that data shared by one company should be invisible and inaccessible to all other companies working with the cluster. That is a key requirement from any business – to prevent illegal industrial espionage and stealing of sensitive data. In this sense, probably the most reliable, flexible and appropriate integration strategy is by implementing a service-oriented architecture (SOA). According to it, the service provides should deploy a wrapping web service (soap-based or rest-based) on top of the entire Hadoop cluster that will allow external authorized users (could be applications or people) to load and analyze data within the cluster.

The aim of this article is to propose an architecture for integration of a company's heterogeneous data (stored within a database of any type, or in the filesystem) to a remote Hadoop cluster, providing data analytical services on demand. Such integration will give the company (business organization) an affordable, cost-effective, access to powerful big data analytical tools as the ones included in the Hadoop Ecosystem. The paper is structured as follows: Section 2 reviews some previous work done by other researchers. Section 3 proposes architecture for integration of a company's data to a remote Hadoop cluster accessible as a service. Finally, Section 4 ends the article with a conclusion, outlining the usefulness of the proposed solution.

II. Related Work

Big data management has introduced some challenges (maintaining heterogeneous data, large data volumes and increased data throughput) for applications, related to data processing. Currently NoSQL databases are proposed to address these challenges by offering horizontal scalability, high availability and data storage without using fixed schemas. NoSQL databases, however, do not have a standard query language, which makes developers' life harder. On the other hand, the traditional relational databases and the SQL language are very popular for processing and storage critical data, but not suitable for large volumes of data.

As a result, multiple approaches have been proposed to define, process and store large volumes of relational data in NoSQL databases by using similar to SQL interfaces. They all focus on both scalability and availability. Schreiner et al. present a comparative analysis [3] of these approaches based on their architectural solutions. The authors motivate their research with the fact that the use of generalized architectural solutions is a proper strategy for relational-based applications, which could be used to move/migrate relational data to NoSQL databases. One of the approaches for that is to propose a way to access NoSQL databases by SQL-like commands. Suggested methods, however, convert the relational data model to a single NoSQL data model and sometimes directly to a specific NoSQL database management system (DBMS). Schreiner et. al. present a canonical approach, called SQLToKeyNoSQL [4], which converts relational schemas and SQL commands to equivalent schemas and data access methods for any NoSQL DBMS, based on the key-value data model, the document data model, and the column-oriented data model. The authors propose architecture of a layer, focusing on strategies for data transformation and mapping.

The analysis of the relational and the non-relational databases leads to the conclusion that these data storage and processing systems are to some extent complementary. Unfortunately, the complementary behavior has a negative effect on the integration possibilities, both in terms of data model and data processing. In terms of performance, it may be useful to rely on multi-lingualism, multi-model approach or multi-step modeling, or even to transform the SQL schema into a NoSQL model and then to migrate the data. Another option is to integrate relational and non-relational databases by the help of a third-party data model. This option is discussed by Pokorný in [5]. Together with that, the author presents some new methods for designing such integrated database architectures. He also discusses that it is not possible to simply apply traditional approaches for integration of relational and non-relational DBMSs, due to the complementary nature of these two types of databases. The author reviews multiple methods for integration of these heterogeneous databases. The heterogeneity itself leads to a new set of problems. NoSQL databases are flexible, but their data design also requires specific modeling solutions that affect their performance in the integrated architectures.

Integration of data stored in heterogeneous systems is a quite challenging task as well and very difficult to solve. Vathy-Fogarassy and Hugyák propose a new data integration methodology [6] in order to provide personalized data queries to multiple relational and NoSQL database management systems. Their solution does not support joins and aggregation of data from multiple sources, but it just collects and migrates data from/to the different individual sources. The method is based on a metamodel approach and covers the heterogeneity of the source systems in terms of their structure, semantic and syntactic features.

Due to the complex management of organizations, the corporate applications could contain a vast number of tables with multiple relationships and constraints among them. Organizations are currently still using relational databases, but the NoSQL systems are constantly increasing their market share due to their excellent performance and high availability. So, more and more migration tools will be needed to migrate relational to non-relational data models. The database schema is important not just for the relational databases, but for the

NoSQL systems as well, since it could provide better query execution. Although the NoSQL databases do not explicitly require database schemas, maintaining such information is important for activities such as data integration, data validation and operational compatibility. Frozza et. al. suggest in [7] a process to automatically extract schemas of NoSQL databases relying on the column-oriented data model. The authors use JSON as a canonical data representation format and test their approach by extracting schemas from HBase. Dai discusses a transformation technique [8] that is capable of transforming database schemas, and translate and optimize queries. Other approaches to design and transform schemas could be found in [9],[10].

Processing and storage of large data volumes by using conventional techniques is not possible. Nowadays, organizations should target their development to non-relational (NoSQL) systems since they support more flexible data models. It is a great challenge for business organizations to transform their existing relational data to NoSQL data models, especially when having in mind the heterogeneity and the complexity of their data. Furthermore, big data management faces an additional challenge – data cleaning and flushing. Ramzan et al. suggest a solution that supports two modules [11] – data transformation module and data cleaning module. The first phase transforms the relational database to a NoSQL database by applying an appropriate transformation model. Then the second phase provides data cleaning, aiming to improve data quality and prepare them for further (big data) analyses.

Most data model transformation methods target a single (specific) NoSQL model and provide a little or no support for transformation's personalization. Kuszera et al. present an approach [12] for transforming relational to NoSQL databases that supports the document-oriented and the column-oriented data models. Their method uses as an input a set of directed acyclic graphs (DAGs) that represent the targeted NoSQL model. DAGs are used to generate data transformation commands. The approach supports different relations' cardinality and the transformation commands could be personalized. The authors developed a tool that interprets the input DAGs and supports multiple transformation strategies.

Since big data processing applications, based on Cloud Computing become more and more popular, many existing systems will upscale their services in order to support the increasing volumes of data. Liao et al. propose a data transformation system [13] that provides hybrid architecture, including both relational and NoSQL databases. It supports simultaneous query processing and data transformation, and data synchronization. The authors focused their work on the transformation speed and the diversity of the data.

Modern applications that process different data formats often use several types of databases, and the need to migrate data between them is common. Although there are multiple methods to perform data model conversion, the process of choosing the ideal data structuring for specific application requirements is not a trivial task. Kuszera et al. describes an approach for converting relational databases to NoSQL ones, consisting of multiple steps [14] for defining, evaluating and comparing alternative NoSQL schemas (data structuring), before migrating the data.

III. Architecture for Integrating a Company's Heterogeneous Data to an Existing Remote Hadoop Infrastructure

The existing approaches, reviewed in the related work, offer an integration of a relational database management system to a non-relational one with a specific target data model. This is important for companies to move their data from an old relational database management system to a modern, flexible NoSQL one, supporting data models suitable for big data processing. However, this data model transformation does not provide any data analytics at all. It just transforms the existing data. In contrast, the architecture proposed here allows integration of any type of data to an existing Hadoop infrastructure, providing powerful data analytical tools.

In order to make the most of the data analytical capabilities of the Hadoop Ecosystem, the company should be able to share not only its database, but also files whose content could be analyzed as well. So, integration is needed on multiple levels:

- Integration with the Hadoop Distributed File System (HDFS).
- Integration with the real-time column-oriented database management system HBase.
- Integration with the Hive service and obtaining aggregated analyses of it.

Integration with HDFS allows the company to share data exported from any type of database managements systems (DBMSs) – relational or NoSQL. Data could be exported in the form of CSV (comma separated values) data files or in other format. Once loaded in HDFS, these files could be later imported in any Hadoop's data analytics service. This approach provides very high level of flexibility and abstraction from the specific DBMS, and allows integration of any type of data to the Hadoop cluster. Integration could be implemented by using HDFS's REST API, called WebHDFS. There is other alternative as well – through the HDFS client distributed together with Hadoop, but the WebHDFS provides higher flexibility.

If it is needed the company's data to be stored in real time within the Hadoop infrastructure, then an integration with the column-oriented database management system HBase [15],[16] is necessary. It could be done through:

- The HBase's REST API.
- Thrift Server.

There is a third way as well – by storing data in files and loading them in HDFS, but it is not suitable for real-time operations.

The HBase's REST API provides an easy access to HBase, and since communication is done over the HTTP protocol, the client could be implemented in any programming language.

The problem however is that both the RESTAPI and the Thrift Server do not provide any access control by default. If a user knows the namespaces and the tables' names of another user, then the first one could easily steal the data of the second user. A possible solution of that is to use the Kerberos authentication protocol [1]. An alternative solution is to develop a wrapping web service around the HBase's REST API (or the Thrift Server) that should handle user authentication and perform access control.

Another useful integration is with the Hive service [1],[2]. Hive is not a database, but just a query engine that allows searching and retrieving data from different data stores by using a language that is almost identical to SQL. It is called HQL – Hive query language. The key point here is that data could be retrieved and analyzed from different data stores. Actually, HQL runs over temporarily virtual tables that could be built from multiple sources – HBase, ordinary files from HDFS and other sources. This is the highest strength of Hive – it allows using SQL to search and process data from files. Usually, files which are mapped as virtual tables are csv and tab-delimited data files, but could be also MS Excel or other sources of structured or semi-structured data. So, Hive could be used to perform data aggregation and analyses of: real database data; system log files; application log files; accounting data files; csv and MS Excel spreadsheets; and many others sources of data. Within the real world, such data files are generated as often as data stored within a database. Furthermore, Hive supports the Hadoop's MapReduce framework, meaning that the HQL queries could be distributed among multiple servers and run in parallel, which allows analysis to be performed on very large amounts of data.

Fig. 1 presents a conceptual architecture for a company's data integration (database and filesystem) to an existing Hadoop cluster. It implies design and development of just two new modules (applications) – "the integration middleware / wrapping service" from the service provider's side and the "Customer's data integration tool" from the client's side. All other components (except DBeaver) are existing services provided by Hadoop or "built-in" applications in the Hadoop Ecosystem.

Modules marked in peach color are the main Hadoop services - HDFS, HBase, Hive and Hue [1],[2]. Modules marked in white within the cluster are accompanying application programming interfaces (APIs) that allows third party applications to interact with the Hadoop services (the modules in peach). These APIs are not directly built-in within the main Hadoop services, but implemented as separate (external) applications, which however are usually distributed together with the services. The easiest way to install all of them is by installing Cloudera Manager that deploys the Cloudera Distributed Hadoop (CDH).

Cloudera Manager offers a user-friendly, comprehensive interface for installation and management of the entire Hadoop Ecosystem. It could be used to start, configure and stop individual services in the cluster. However, if services are installed as stand-alone applications, then the APIs should be started from a command-line interface and configured through xml files.

DBeaver is a database client, in general, that could connect to multiple relational and non-relational DBMS, and to some services (including Hive) from the Hadoop Ecosystem as well. It provides a graphical user interface to execute SQL queries (and not only) and visualize the results. It can be used directly by end users, without any skills in system administration and programming.

The company's database could be of any type – relational, non-relational, or even file-based. Its data are retrieved based on the database type and then packed within the necessary data structures, which are sent to the "Integration Middleware" for further processing and storing within the cluster.

The two custom and important modules are "Integration Middleware / Wrapping Service" and "Customer's data integration tool". They are the ones, which are directly responsible for data integration. In contrast to the others, they do not exist in advanced, but should be designed and implemented entirely for the purpose of integration.

1) "*Customer's data integration tool*": provides a graphical user interface (GUI) that allows the company to manage the data it wants to integrate (including to choose what exactly to share) with the Hadoop cluster, and also to render/visualize the received results. This module exactly extracts the data from the local database and packs them in suitable data structures (usually JSON objects) that are later send to the "Integration Middleware / Wrapping Service". The module should work with any type of database management systems (relational or non-relational) and pack the data in similar manner, regardless of their type. The customer's data integration tool should allow browsing not only the local database, but the local filesystem as well. User should be able to browse, select and upload files to the remote HDFS filesystem for further processing and analysis by Hadoop's data analytics services.

2) "*Integration Middleware / Wrapping Service (API)*": implements data integration on the service provider's side. It works as a wrapping service around the entire Hadoop cluster. The customer's data integration tool works with this module only, not directly with the Hadoop services or APIs. In this sense "Integration Middleware / Wrapping Service" *serves as a mediator* between the company (the client) and the WebHDFS API, HBase API and the Thrift Interface. But why such a mediation service is needed? Because all the three interfaces do not provide any user authentication and access control. For HBase, for example, if one user knows the namespaces and table names of another user, then the first one could steal the data of the second one. Similar apply to the Thrift interface as well – all connections to HBase are done through the same socket without any authentication. All users have access to everything, by default. And that is a big problem. No company will share its data if they become publicly available and easily accessible to competitors. To solve this problem the Kerberos authentication service could be installed and configured, or to implement a wrapping service around the APIs to perform user authentication and

access control. The latter could be easily integrated in this module and perform a flexible, custom access control.

The second important task of this module is to convert the data from their source model (relational or whatever) to column-oriented one, suitable for loading in HBase. This is the module that is solely responsible for data conversion. They arrive from the "Customer's data integration tool" packed as JSON objects, then they should be converted to suitable model and loaded to HBase through the API or the Thrift Interface.

3) Conversion of source data models to the column-oriented data model suitable for HBase: As already mentioned, conversion from a source data model to a target data model is important for data integration from heterogeneous systems. Research in data models transformation has intensified a lot during the last years – multiple approaches have been proposed, as seen from the related work section. Most of them apply the relational model as a source model, since there are many legacy data storage systems currently in use that relies mainly on the relational model.

In contrast to the traditional relational databases, which are strictly normalized and should meet the requirements of the 3-rd and the BCNF normal form, almost all NoSQL data stores are actually not normalized at all – data could be grouped and replicated in order to increase the read speed. Although the NoSQL databases also maintain some consistency, there may be inconsistencies in the data for a short period of time after performing an operation. As a result, the NoSQL databases are mainly suitable for OLAP systems, but not for OLTP systems, where support of the ACID properties is mandatory.

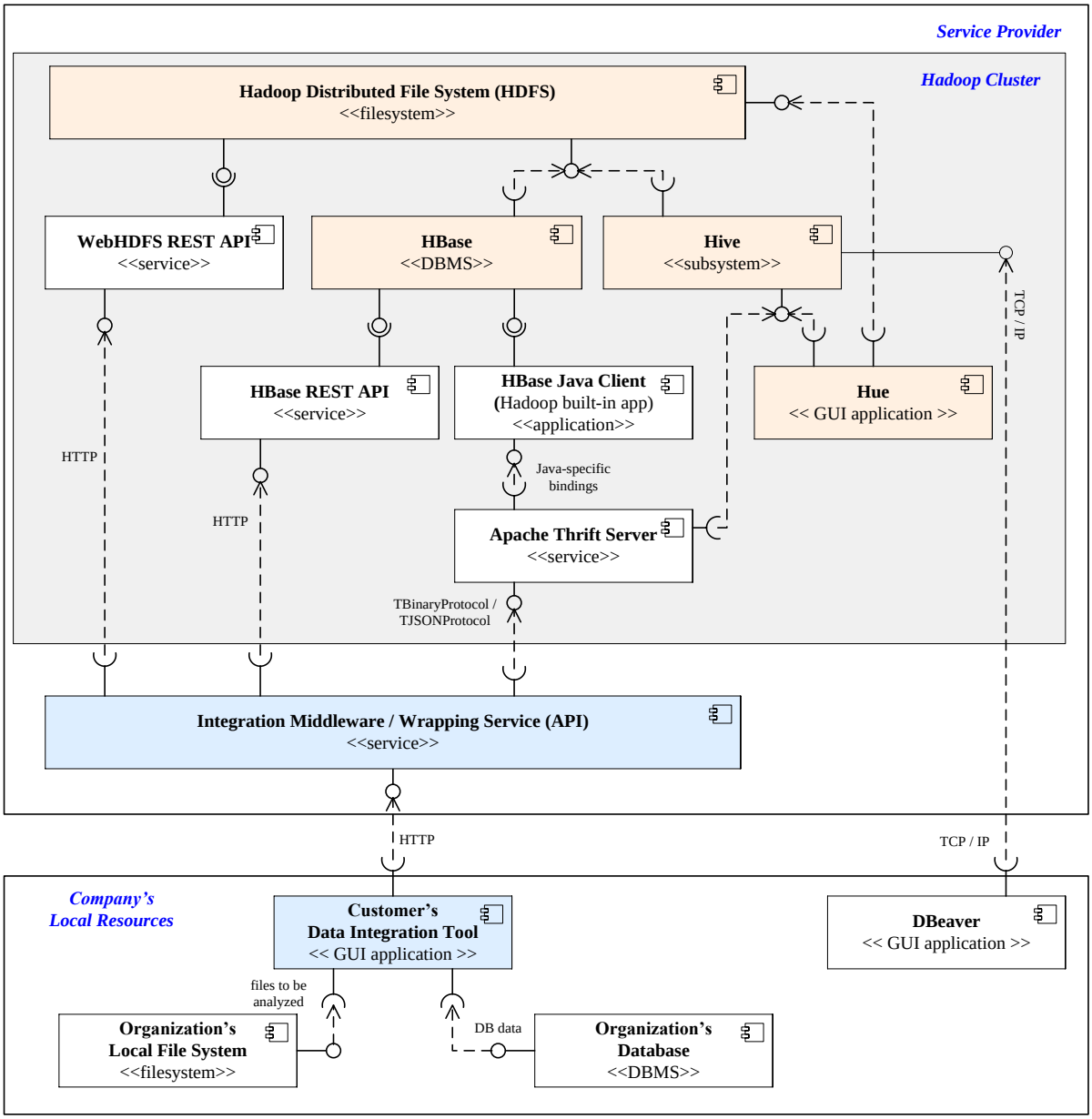

Fig. 1. Architecture for Integration of a Company's Database and Filesystem to Hadoop Infrastructure.

Any data model could be transformed to any non-relational model by applying the methods and algorithms described in [17],[18],[19],[20]. Since this article is related to data integration with an existing Hadoop cluster, the target model here is the column-oriented data model, used by Apache HBase [15],[16]. Here is an intuitive heuristic approach for transforming the relational data model to a column-oriented one:

1) Grouping all related data in a single column family. These are data that should be read/written from/into the database at once. These are the tables' records (rows) in the relational databases. Actually, the column-oriented schema could be easily extracted from the relational schema, since the latter provides more than enough information about attributes' relationships. Every table in the relational database becomes a table in the column-oriented database, and the primary keys of the former are used as row keys in the latter. All attributes within the relational table are group together in a single column family, in order to guarantee that they will be stored and read together.

2) Convert the relationships between tables in the relational schema to "relationships" in the column-oriented schema. There are three types (cardinalities) of relationships in the relational schema: one-to-one; one-to-many; and many-to-many. They will be reviewed separately since the conversion is done in different way:

a) One-to-one. This is intuitive – a column with a foreign key is treated is an ordinary column and could be grouped together with other columns, based on rule 1.

b) One-to-many. In the column-oriented data model, this relationship is actually implemented in the "opposite way" in comparison to the relational model. In the latter, the foreign keys (multiple values) are placed at the side of "many" of the relationship, since the database should meet 1-st and 2-nd normal form. However, in the column-oriented model, multiple values could be grouped together in a single column family. So, to create a one-to-many relationship, a new column family should be created at the side of "one", which could keep multiple values, pointing to the keys at the side of "many". In other words, in the column-oriented model, all the foreign keys are kept at the side of "one".

c) Many-to-many. Implementing this relationship is even easier in the column-oriented data model. In the relational model, many-to-many relationships are created by the help of a third (called "connection") table that keeps and combines the foreign keys of the other two tables. However, in the column-oriented model, a single column family can contain multiple values. So, to create a many-to-many relationship, a new column family needs to be created at each table, containing the values of the row keys of the other table. There is no need of a third table at all.

In the relational model, the relational database management system (RDBMS) guarantees data integrity and relationships integrity. However, in the column-oriented data model, this is not the case, so the application itself, which uses the data, should be designed to check, maintain and validate data integrity. This responsibility is moved from the DBMS to the software developer. As a result, it is possible to have data inconsistencies for a short period of time, until the next read.

IV. Conclusion

Data analysis is very important for the development of business organizations. It helps them to identify organizational bottlenecks, optimize business processes and increase efficiency and profits. In the era of big data, their analysis could not be done manually. There are powerful data analytical software tools, but they are either expensive or hard to deploy and requires multiple high-performance servers to run. Buying expensive hardware and software, and hiring high-qualified IT experts, is not affordable for all companies, especially for smaller ones and start-ups.

The architecture proposed in this article allows integration of a company's heterogeneous data (both database and filesystem) to a remote Hadoop infrastructure, providing powerful data analytical services on demand. This is an affordable and cost-effective cloud-based solution, suitable for any type company, from start-ups to large-size companies. Businesses are not required to buy any (expensive) hardware or software, but use the data analytical services on demand, paying a small processing fee per request or by subscription. The architecture could be easily implemented by developing just two data integration modules – an integration middleware or a wrapping service around the Hadoop data analytics services, at the service provider's side; and a graphical user interface (GUI) to work with the integration middleware, at the company's side.

The major difference between the proposed architecture and the existing approaches, reviewed in the related work, is that they offer an integration of a relational database management system to a non-relational one with a specific target data model. Instead, our architectural approach provides integration of data of any type (including relational, non-relational and even ordinary text files) to an existing Hadoop cluster. The purpose of this integration is to provide companies with cost-effective and easy to use access to the Hadoop's data analytical tools, not to permanently convert their data to other data models.

Since the proposed architecture will be very useful to the business organizations, providing them an easy access to powerful data analytical tools, and is fairly easy to deploy in the same time, our next goal is to implement it and prove its real usefulness and effectiveness. In general, it allows integration of any type of data to an existing Hadoop infrastructure. However, for this to happen, the source data model should be recognized and data should be parsed properly before sending them to the "Integration Middleware" module. For that reason, we plan to introduce data type documents (DTDs) that contain a formal definition of the source data models and the relevant parsers for these DTDs. Then the integration of a new, even custom, source data model will be quite easy – just by uploading the associated DTD. The use of DTDs highly increase flexibility of the proposed approach and allows integration of virtually any type of data.

ACKNOWLEDGMENT

This paper is supported by project 2022–EEA–01 "Analysis of big data processing algorithms and their application in multiple subject domains", funded by the Research Fund of the "Angel Kanchev" University of Ruse.

REFERENCES

[1] A. Holmes, Hadoop in practice, Manning Publications Co., 2015.

[2] B. Lublinsky, K. Smith, A. Yakubovich, Professional Hadoop solutions, John Wiley & Sons, Inc., 2015.

[3] G.A. Schreiner, D. Duarte and R.D. Mello, "When Relational-Based Applications Go to NoSQL Databases: A Survey", Information, 10, 241, 2019.

[4] G.A. Schreiner, D. Duarte and R.D. Mello, "Bringing SQL databases to key-based NoSQL databases: a canonical approach," Computing, vol. 102, pp. 221-246, 2019.

[5] J. Pokorný, "Integration of Relational and NoSQL Databases", Vietnam Journal of Computer Science, vol. 6, no. 4, pp. 389-405, 2019.

[6] A.Vathy-Fogarassy and T. Hugyák, "Uniform data access platform for SQL and NoSQL database systems", Information Systems, vol. 69, pp. 93-105, 2017.

[7] A.A. Frozza, E.D. Defreyn and R. dos Santos Mello, "An Approach for Schema Extraction of NoSQL Columnar Databases: the HBase Case Study", Journal of Information and Data Management, vol. 12, no. 5, pp. 384–395, 2021.

[8] J. Dai, "SQL to NoSQL: What to do and How", IOP Conference Series: Earth and Environmental Science, vol. 234, 2019.

[9] K. Herrmann, H. Voigt, T.B. Pedersen and W. Lehner, "Multi-schema-version data management: data independence in the twenty-first century," The VLDB Journal, vol. 27(4), pp. 547-571, 2018.

[10] M.J. Mior, K. Salem, A. Aboulnaga, R. Liu, "NoSE: Schema design for NoSQL applications," In Proc. of the 32nd International Conference on Data Engineering (ICDE), pp. 181–192, 2016.

[11] S, Ramzan, I.S. Bajwa, B. Ramzan, and W. Anwar, "Intelligent Data Engineering for Migration to NoSQL Based Secure Environments", IEEE Access, vol. 7, pp. 69042-69057, 2019.

[12] E.M. Kuszera, L.M. Peres and M.D. Fabro, "Toward RDB to NoSQL: transforming data with metamorfose framework", in Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019.

[13] Y. Liao, J. Zhou, C. Lu, S. Chen, C. Hsu, W. Chen, M. Jiang and Y. Chung, "Data adapter for querying and transformation between SQL and NoSQL database", Future Generation Computer Systems, vol. 65, pp. 111-121, 2016.

[14] E.M. Kuszera, L.M. Peres, and M.D. Fabro, "Exploring data structure alternatives in the RDB to NoSQL document store conversion process", Information Systems, vol. 105, 2022.

[15] M. Kerzner, S. Maniyam, HBase Design Patterns, Packt Publishing Ltd., 2014.

[16] R. Choudhry, HBase High Performance Cookbook, Packt Publishing Ltd., 2017.

[17] L. Stanescu, M. Brezovan and D. Dan Burdescu, "Automatic mapping of MySQL databases to NoSQL MongoDB." In Proc. of 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 837-840, 2016.

[18] M.C.d. Freitas, D. Souza and A. Salgado, "Conceptual mappings to convert relational into NoSQL databases," in: Proc. of the 18th International Conference on Enterprise Information Systems, ICEIS, pp. 174–181, 2016.

[19] R. Ouanouki, A. April, A. Abran, A. Gomez and J.M. Desharnais, "Toward building RDB to HBase conversion rules," Journal of Big Data, vol. 4(1), pp. 1-21, 2017.

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