Towards a Data Warehouse Architecture for Managing Big Data Evolution

Darja Solodovnikova and Laila Niedrite
Faculty of Computing, University of Latvia, Raina blvd. 19, Riga, Latvia

Keywords: Data Warehouse, OLAP, Big Data, Evolution, Adaptation, Architecture.

Abstract: The problem of designing data warehouses in accordance with user requirements and adapting its data and schemata to changes in these requirements as well as data sources has been studied by many researchers worldwide in the context of relational database environments. However, due to the emergence of big data technologies and the necessity to perform OLAP analysis over big data, innovative methods must be developed also to support evolution of data warehouse that is used to analyse big data. Therefore, the main objective of this paper is to propose a data warehousing architecture over big data capable of automatically or semi-automatically adapting to user needs and requirements as well as to changes in the underlying data sources.

1 INTRODUCTION

Data warehousing and OLAP technologies have been used to analyse data stored in relational databases to support decision making for several decades. During this time, the majority of research problems related to traditional data warehousing have been solved. However, during the course of time, data that should be processed and analysed in the decision-making process have become so enormous and heterogeneous that traditional solutions based on relational databases have become unusable to process all these data volumes. Besides, new data sources (bio-medical data, social networks, sensor data, log files, etc.) that may be used for decision making have emerged. In most cases, data obtained from such sources are semi-structured or unstructured, therefore special methods must be applied to store, process and analyse them.

The increase of volume, variety and velocity of data to be stored and processed induced the emergence of new big data technologies that employ distributed data storage techniques where special methods are used to process data in parallel. The demand to analyse data stored in such systems is increasing and one of the analysis options is to use data warehousing and OLAP methods. Traditional methods applied in relational databases may not be used anymore for big data analysis, therefore, new modern technologies (Shvachko et al., 2010), (Zaharia et al., 2010), tools (George, 2011), (Olston et al., 2008) and frameworks (White, 2009) have been proposed recently to support Big Data analytics, including OLAP-like analysis (Thusoo et al., 2010), (Apache Kylin Overview, n.d.). These tools mainly support just the data warehouse maintenance when volume of the underlying source data is growing, but the problem of data expansion when the structure of data evolves or new data items are added to data records remains an unsolved yet challenging task. Besides, to handle changes in user requirements, a great developer effort must be devoted to manually adapt the system to underlying changes, inasmuch as the existing solutions do not support automatic or semi-automatic handling of evolution of a data warehouse over big data.

Since big data is a new technology, there are a number of research directions and challenges that have been outlined in several recent articles (Kaisler et al., 2013), (Cuzzocrea et al., 2013), (Abaker et al., 2015). The authors of the paper (Kaisler et al., 2013) mention dynamic design challenges for big data applications, which include data expansion that occurs when data becomes more detailed. Another review paper (Cuzzocrea et al., 2013) is more specific and indicates research directions in the field of data warehousing and OLAP. Among others, the authors also mention the problem of designing OLAP cubes according to user requirements.

Solodovnikova, D. and Niedrite, L.
Towards a Data Warehouse Architecture for Managing Big Data Evolution.
DOI: 10.5220/0006886100630070
In *Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018)*, pages 63-70
ISBN: 978-989-758-318-6

To address the evolution problem, we propose a system architecture for warehousing and performing various kinds of analysis, including OLAP analysis, over structured and unstructured big data at different granularity levels that are loaded into the system with different velocity. The unique feature of our proposed architecture is that it is capable of automatically or semi-automatically adapting to changes in requirements or data expansion.

The rest of the paper is organized as follows. In Section 2 the related work is discussed. The main contribution of this paper is presented in Section 3, where the data warehouse architecture is described and the process of handling evolution in the proposed architecture is outlined. We conclude with directions for future work in Section 4.

2 RELATED WORK

The problem of data warehouse evolution has been studied extensively in relational database environments. The approaches for handling evolution at the extraction, transformation and loading (ETL) level is proposed in the paper (Wojciechowski, 2018). Schema evolution approaches, e.g., (Bentayeb et al., 2008), update data warehouse schemata to reflect changes that have occurred. Another direction is schema versioning and temporal data warehouses. According to these approaches, not only schema of a data warehouse is updated, but also the history of previous schemata is kept, e.g., (Golfarelli et al., 2006), (Ahmed et al., 2014), (Malinowski and Zimányi, 2008). These works do not describe formally how the information requirements affect the evolution changes. Some research is done to get such methods (Thenmozhi and Vivekanandan, 2014), (Thakur and Gosain, 2011).

All the above-mentioned approaches target data warehouses implemented in relational database environments, thus, they cannot be utilized directly to perform adaptation of big data warehouses. Only few big data solutions are considering the evolution aspect of big data. The paper (Olston et al., 2011) presents a workflow manager deployed at Yahoo. The paper also explains the approach to data schema specification and schema evolution processing. The objective of the presented system is not data analysis and it does not employ data warehouse.

A solution to handling data source evolution problems in the integration field was presented in the paper (Nadal et al., 2017). The authors propose the use of big data integration ontology to define the integrated schema, source schemata, their versions and local-as-view mappings between them. When a change at a data source occurs, the system steward supplements the ontology with a new release and a new wrapper that allows unchanged and new attributes to be added to the changed source. Our approach differs in that the proposed architecture is OLAP-oriented and is capable of handling not only changes in data sources, but also requirements.

There are several studies devoted to multidimensional analysis of big data. The paper (Song et al., 2015) presents an OLAP system for big data implemented with Hadoop. The authors propose a data model utilized in the system and algorithms for execution of aggregate queries, roll-up and drill-down operations and for data distribution among multiple nodes of the system. The solution leverages "share-nothing" architecture, which consists of a Hadoop cluster where OLAP cubes are calculated by an ETL tool, metadata server, job node for query management, OLAP service facade and OLAP client used for query specification and visualization of query results.

A distributed OLAP system is presented in the paper (Chen et al., 2017). The system architecture is composed of 4 modules. Data acquisition module is responsible for obtaining data from structured and unstructured data sources. Data storage module maintains source data in HDFS, key-value store and relational database. OLAP analysis module performs OLAP cube calculation and SQL-like query execution. Data visualization module allows definition of OLAP cubes to be calculated, visualization of query results and specification of user privileges.

The paper (Santos et al., 2017) studies the problem of migration of a traditional data warehouse to a data warehouse over big data. The authors propose a big data warehousing architecture where data from data sources are obtained by ETL tools and initially accumulated in the HDFS staging area, then data are transformed and loaded into the big data warehouse implemented in Hive. Data analytics and visualization is available in Tableau that uses Impala for querying data warehouse data.

All the previously discussed studies on data warehouses for big data analysis do not address the problem of big data evolution. We found only a few researches that consider that problem. One of them is the paper (Tardio et al., 2015) that proposes a methodology for a construction of a system for big data analysis. One of the methodology steps is a design of a multidimensional model, which involves analysis of data sources, construction of separate

models for each source and data integration into a unified multidimensional model. The authors also discuss a model enrichment process, which may take place when a new data source is added to the system or additional data are discovered by means of data mining methods. To handle changes in big data, the authors propose iterative execution of the model design step until such a model is obtained that can satisfy all information requirements. The proposed approach is complimentary to ours, however, it cannot be applied in case when some previously available data become unavailable.

A data warehouse solution for big data analysis that is implemented using MapReduce paradigm is described in the paper (Chen, 2010). In this solution, virtual views that join dimensions and fact tables of a data warehouse are built and processed by SQL-like queries. The presented system is used for advertisement data analysis and the author mentions that because of the system nature, schema changes in it occur frequently. Therefore, the system supports two kinds of changes: slowly changing dimensions are managed with methods proposed by Ralph Kimball (Kimball and Ross, 2013) and fact table changes are handled by schema versions in metadata. Unlike our proposal, the system does not process changes in big data sources that may significantly influence analysis process and results.

An architecture that exploits big data technologies and is used for large-scale data processing and OLAP analytics at LinkedIn is presented in the paper (Sumbaly et al., 2013). The architecture supports also source data evolution by means of maintaining a schema registry and enforcing the schema to remain the same or compatible to the desired structure. OLAP functionality is provided by the distributed system Avatara described in the paper (Wu et al., 2012). Avatara consists of two sub-systems: batch processing engine for sharded small OLAP cube calculation and incremental update implemented in Apache Hadoop and online SQL-like query engine. Such an architecture provides that user-defined queries are executed with very low latency, but data used for queries are updated less frequently. Authors mention also the problem of cube evolution when a new dimension is added to the cube. In contrast to our proposed architecture, in Avatara evolution is handled by means of manual redefinition of cube schema and cube data re-calculation.

3 BIG DATA WAREHOUSE ARCHITECTURE

To solve the topical issue of adaption of a data warehouse over big data to changing user requirements and expanding data sources, we propose the use of a big data warehousing architecture to analyse data stored and processed by big data technologies. The architecture is an adapted version of the data warehouse evolution framework presented in (Solodovnikova, 2007). The data warehouse evolution framework is capable of automatically detecting and handling changes in data sources or a traditional data warehouse as well as adapt a data warehouse to changes in user requirements (Solodovnikova et al., 2015) utilizing a schema versioning approach.

The architecture proposed in this paper will be able to provide big data analysis capabilities and handle big data evolution. The big data warehouse architecture is depicted in Figure 1.

3.1 Components of the Architecture

The architecture consists of several components: source layer, data highway, metastore, cube engine, metadata management tool and adaptation component. The functionality of each component is described in the following subsections.

3.1.1 Source Layer

In the source layer, data from various heterogeneous data sources are obtained and loaded into the system for further analysis. Because of the variety of big data that may be required for the analysis, the proposed architecture must process not only data structured in rows and columns, but also semi-structured data, for example, in XML or JSON formats, and even unstructured data, for example, log file records, user posts, photographs or video.

In the architecture, wrappers obtain data from data sources and load them into the system. Wrappers implement interfaces for data acquisition supported by corresponding data sources (for example, web services, API, etc.). Data are coming into the system at different rates. For example, data from RDBMS may be loaded in batches, but sensor measurements or log file records are incoming as a stream. All data are loaded in their original format into the raw source data storage of the data highway explained in the next subsection.

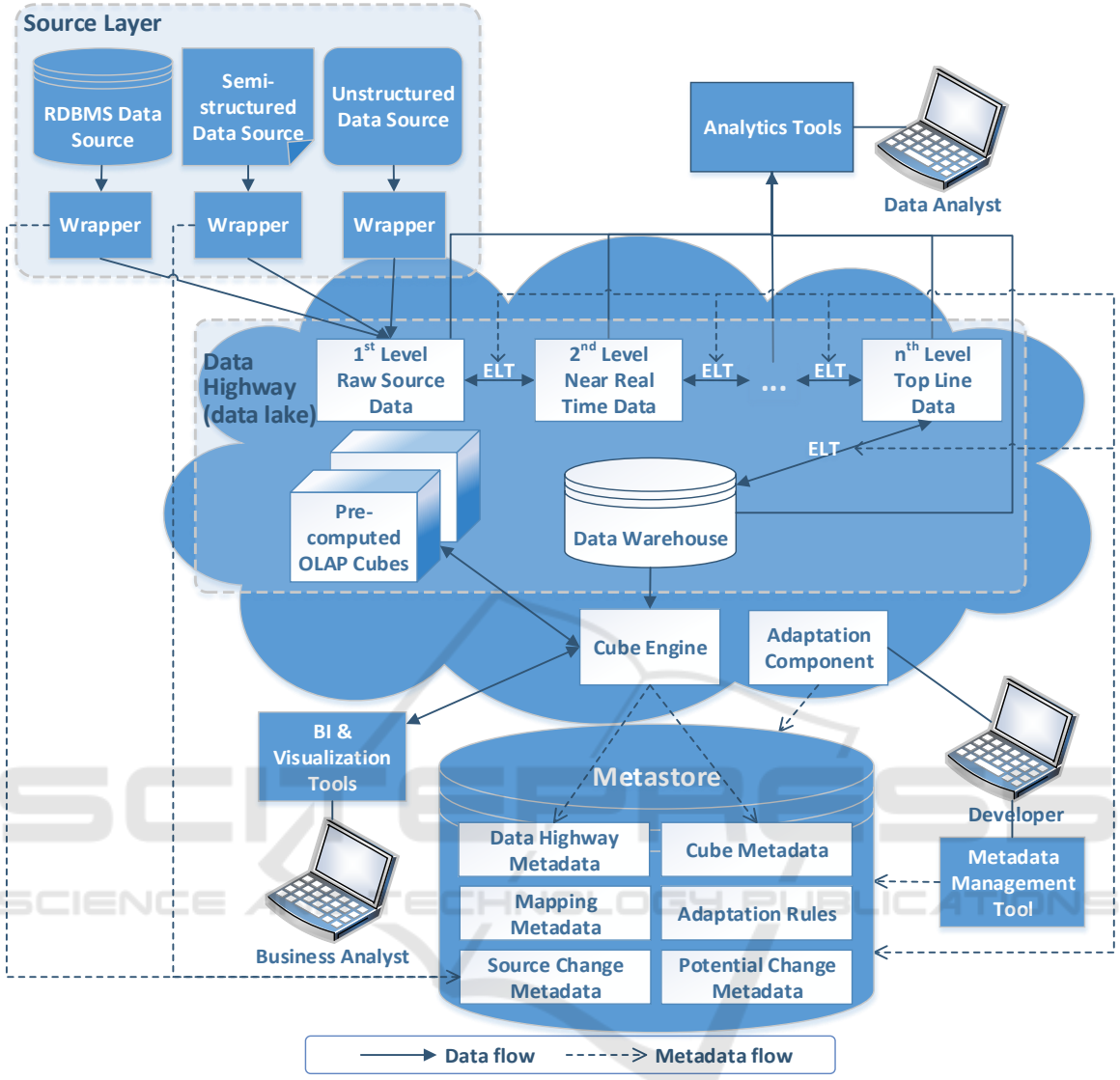

Figure 1: Big data warehouse architecture.

3.1.2 Data Highway

In the big data warehouse architecture, we adopt the idea behind the architecture best practices for big data proposed by (Kimball & Ross, 2013) to build a highway of data at different levels of latency. Starting from the raw source data, each next level data are obtained from the previous level data and are updated less often. Apart from different update frequency, the latter level data from multiple heterogeneous sources are integrated and aggregated, so to support lower query latency. The number of levels and their update frequency is determined by requirements of a particular system.

Finally, source data are transformed into a structured data warehouse schema (star or snowflake). Since data in the raw source level and intermediate levels of the highway are stored in the original format and are only fairly integrated, we propose to utilize a data lake repository (Terrizzano et al., 2015) for storage and management of these data.

Before being loaded into each next level of the highway and finally a data warehouse, raw source data need to be transformed. Since data in the proposed architecture are firstly copied in their original format and transformed at the later stage, ELT (Extract, Load, Transform) rather than ETL processes (Cohen et al., 2009) are responsible for pre-processing data incoming from heterogeneous data sources. Existing metadata-oriented ETL methods based on mappings of source elements to

data warehouse elements may be utilized in case of structured or semi-structured data sources loaded in batch mode. In contrast, data incoming from unstructured data sources need to be pre-processed quite highly. For example, data mining or sentiment analysis techniques may be applied on them to gain structured information that is later to be loaded into the data warehouse.

After being loaded into the data warehouse, the data at each level of the highway are supplemented by the generated surrogate keys of dimensions to ensure data provenance and enable handling of evolution. Since ELT processes augment data at the lower level by information obtained during the processing and transformations performed at the higher level, it is possible to join data from different levels of the data highway to perform a more valuable analysis.

3.1.3 OLAP Cube Computation

The main objective of the architecture is to support OLAP analysis of big data. Since the volume of data stored in the data warehouse may be too large to provide a reasonable performance of data analysis queries, it is possible to pre-calculate certain OLAP cubes and execute queries on pre-computed data sets. To speed up queries, the cube engine component of the architecture pre-computes various dimensional combinations and aggregated measures for them according to the developer specification and saves computation results in the distributed storage. Furthermore, the cube engine manages execution of queries by routing the query to one of these pre-computed cubes.

3.1.4 Metadata Management

The operation of the architecture is highly based on the metadata in the metastore. The metadata management tool is used by a developer to define metadata in the metastore. The metastore incorporates six types of interconnected metadata. Data highway metadata store information about the schemata of each level of the data highway as well as of the data warehouse.

Cube metadata are used to specify the dimensions, measures and other metadata of pre-computed cubes. This information is used during the pre-computation process as well as for execution of queries.

Mapping metadata define the logics of ELT processes. They store the correspondences between data obtained from the sources and data items of the data highway as well as necessary transformations that must be made during the data loading process.

Information about changes in data sources is accumulated in the source change metadata. Such information may be obtained from wrappers or during the execution of ELT processes.

Adaptation rules specify adaptation options that must be implemented for different types of changes. Finally, the metastore also includes the potential change metadata, which accumulate proposed changes in the data warehouse schema.

To maintain the information in the metastore, a developer utilizes the metadata management tool. In addition, the metadata management tool allows the developer to initiate changes in the data highway and ELT procedures to handle new or changed requirements for data. The history of chosen changes that are implemented to propagate evolution of data sources, as well as changes performed directly via the metadata management tool, are also maintained in the potential change metadata.

3.1.5 Adaptation Component

The core element of the big data warehouse architecture responsible for handling changes in data sources and information requirements is the adaptation component. The main idea of the adaptation component is to generate several potential changes in a data warehouse or other levels of the data highway for each change in a data source and to allow a developer to choose the most appropriate change that must be implemented. To achieve the desired functionality, the adaptation component uses metadata from the metastore.

To implement certain kinds of changes, additional data may be necessary that cannot be identified automatically, for example, transformations for missing properties of source data records. In such a case, these data are supplied by the developer via the adaptation component and are saved in the adaptation rules in the metastore. The process of handling changes in data sources by the adaptation component is detailed in subsection 3.2.2.

3.1.6 Data Analysis

In our proposed architecture, we plan to support different kinds of analysis. OLAP cubes may be explored by business analysts in the form of dashboards, charts or other reports and by performing OLAP operations using business intelligence and visualization tools. In addition, experienced data analysts may apply advanced analysis techniques (for example, data mining) to

gain insight into data or discover new information useful for decision-making by means of utilizing existing analytics tools or implementing ad-hoc analysis procedures.

3.2 Handling Evolution

The proposed architecture is able to handle source changes that can influence any level of the data highway as well as changes in requirements for data necessary for analysis.

3.2.1 Changes in Requirements

Because data lake repositories allow to flexibly store heterogeneous data in their original format incoming from various data sources, such repositories are very suitable to provide new analysis opportunities that may become required in the course of time. In the big data warehouse architecture, we plan to support such changes in requirements that imply addition of new data to the system, as well as removal of no longer required data. The examples of supported changes include an addition of a new dimension attribute or measure to the data warehouse schema, addition of a new property to data records of an intermediate level of the data highway or removal of such attributes, measures or properties.

All changes in requirements must be handled by the metadata management tool. A developer must make changes in the data highway metadata so to specify new data items that are necessary for analysis or to remove no longer required data items from the metadata. The history of changes made to the schemata of each updated level of the data highway is also preserved. Furthermore, the developer must make changes to the ELT procedures affected by the changes in requirements. The information about correspondences between data items of the sources and new or modified data items of each level of the data highway must be also updated in the mapping metadata to continue supporting handling of possible changes in data sources. Finally, changes in the schema of the data warehouse must be also propagated in the cube metadata so that cube engine could re-create OLAP cubes according to the changed requirements.

3.2.2 Changes in Data Sources

Data are incoming in the system from multiple heterogeneous data sources that can change independently. Many of such changes can invalidate existing ELT processes as well as data required for the analysis. Thus, it is necessary to handle changes in data sources automatically or semi-automatically.

To implement the adaptation of ELT processes and possible schemata of the data highway levels affected by the change, source wrappers must be capable of tracking changes in views of source data provided by interfaces. This functionality may be unavailable, especially in case of unstructured data sources. Therefore, certain changes in data sources can be discovered only during the execution of ELT processes by means of verification of source data correspondence to the desired format and constraints and discovering any inconsistencies caused by changes in the data structure. All changes discovered by wrappers or ELT processes must be registered in the source change metadata. The list of supported changes in data sources is highly dependent on the ability of the corresponding wrappers or ELT processes to track changes.

The processing of source changes is performed by the adaptation component. Initially it analyses the changes in the source change metadata and detects changes that affect the schemata of the data highway and ELT processes. The adaptation component handles these changes using data from the adaptation rules available in the metastore and provided by the developer and for each change generates potential solutions that adapt the schemata of the data highway and ELT processes. The adaptation may affect one or several levels of the data highway, including a data warehouse.

The developer is informed about all discovered changes and their potential adaptation options, so then he/she may approve the most suitable solutions that are later implemented by the adaptation component. Such approach involves developer effort only occasionally when changes occur and does not require a developer to manually adapt ELT procedures and schemata, however it also minimizes the possibility of errors since no adaptation options are implemented automatically and they all need to be approved by a developer.

If schemata of the data warehouse are also adapted after implementing the chosen solutions, the adaptation component must also update cube metadata to reflect changes in the data warehouse and launch the cube engine to re-compute cubes after re-definition.

3.3 Potential Implementation

For the implementation of the big data warehouse architecture, we intend to utilize the existing tools and technologies as well as to implement the

original solutions. We plan to use HDFS for storage of raw source data and intermediate levels of the data highway and Apache Hive for a data warehouse. Apache Kylin will be suitable to implement the functionality of the cube engine since it is capable of producing cubes from Hive data structures. Cubes pre-computed by Kylin need to be stored in Apache HBase database. We plan to employ a RDBMS for the metastore and possibly Pentaho software to implement ELT processes. Finally, since no existing products support big data evolution to the full extent, we will implement the original solutions for the metadata management tool and the adaptation component.

4 CONCLUSIONS

The main contribution of this paper is a data warehouse architecture that on one hand allows to perform different kinds of analytical tasks, including OLAP-like analysis, on big data loaded from multiple heterogeneous data sources with different latency. On the other hand, our proposed architecture is capable of processing changes in data sources as well as evolving analysis requirements. We described the components of the architecture, the necessary metadata and gave examples of changes that are supported by the architecture together with their implementations within the architecture.

Our future research directions include a construction of metadata models to describe schemata of the data highway, requirements for data, source data and changes. The main challenge here would be to determine metadata of non-structured or semi-structured big data and the possible solution is to leverage meta-learning.

To enable handling of big data evolution, algorithms for automatic and semi-automatic change detection and treatment are necessary. Since change detection may be impossible at the data source layer, the possible solution would be to specify constraints on data items incoming from data sources and detect violation of such constraints to discover evolution.

ACKNOWLEDGEMENTS

This work has been partly supported by the European Regional Development Fund (ERDF) project No. 1.1.1.2./VIAA/1/16/057 “Handling Adaptation of Big Data Warehouse” and by University of Latvia project No. AAP2016/B032 “Innovative information technologies".

REFERENCES

Abaker, I., Hashem, T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U., 2015. The rise of “big data” on cloud computing: Review and open research issues. In *Information Systems*, 47(C), pp. 98-115.

Ahmed, W., Zimányi, E., Wrembel, R., 2014. A Logical Model for Multiversion Data Warehouses. In *16th International Conference on Data Warehousing and Knowledge Discovery*, pp. 23-34.

Apache Kylin Overview [Online]. Available at: http://kylin.apache.org (Accessed: 27 April 2018).

Bentayeb, F., Favre, C., Boussaid, O., 2008. A User-driven Data Warehouse Evolution Approach for Concurrent Personalized Analysis Needs. In *Integrated Computer-Aided Engineering*, 15(1), pp. 21-36.

Chen, S., 2010. Cheetah: A High Performance, Custom Data Warehouse on Top of MapReduce. In *VLDB Endowment*, 3(2), pp. 1459-1468.

Chen, W., Wang, H., Zhang, X., 2017. An optimized distributed OLAP system for big data. In *2nd IEEE International Conference on Computational Intelligence and Applications*, pp. 36-40.

Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C., 2009. MAD Skills: New Analysis Practices for Big Data. In *VLDB Endowment*, pp. 1481-1492.

Cuzzocrea, A., Bellatreche, L., Song, I., 2013. Data Warehousing and OLAP over Big Data: Current Challenges and Future Research Directions. In *16th international workshop on Data warehousing and OLAP*, pp. 67-70.

George, L., 2011. *HBase: the definitive guide*. O’Reilly Media Inc.

Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G., 2006. Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. In *Data & Knowledge Engineering*, 59(2), pp. 435-459

Kaisler, S., Armour, F., Espinosa, J.A., Money, W., 2013. Big Data: Issues and Challenges Moving Forward. In *46th Hawaii International Conference on System Sciences*, pp. 995-1004.

Kimball, R., Ross, M., 2013. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. John Wiley & Sons, Inc., 3rd edition.

Malinowski, E., Zimányi, E., 2008. A Conceptual Model for Temporal Data Warehouses and Its Transformation to the ER and the Object-Relational Models. In *Data & Knowledge Engineering*, 64(1), pp. 101-133.

Nadal, S., Romero, O., Abelló, A., Vassiliadis, P., Vansummeren, S., 2017. An integration-oriented ontology to govern evolution in big data ecosystems.

In *Workshops of the EDBT/ICDT 2017 Joint Conference*.

Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A., 2008. Pig Latin: A Not-So-Foreign Language for Data Processing. In *2008 ACM SIGMOD international conf. on Management of data*, pp. 1099-1110.

Olston, C., Chiou, G., Chitnis, L., Liu, F., Han, Y., Larsson, M., Wang, X., 2011. Nova: continuous Pig/Hadoop workflows. In *ACM SIGMOD International Conference on Management of data*, pp. 1081-1090.

Santos, M.Y., Martinho, B., Costa, C., 2017. Modelling and implementing big data warehouses for decision support. In *Journal of Management Analytics*, 4(2), pp. 111-129.

Shvachko, K., Kuang, H., Radia, S., Chansler, R., 2010. The Hadoop distributed file system. In *MSST'2010, IEEE 26th Symposium on Mass Storage Systems and Technologies*. pp. 1–10.

Solodovnikova, D., 2007. Data Warehouse Evolution Framework. In *Spring Young Researchers Colloquium on Database and Information Systems SYRCoDIS*, pp. 4.

Solodovnikova, D., Niedrite, L., Kozmina, N., 2015. Handling evolving data warehouse requirements. In *19th East-European Conference on Advances in Databases and Information Systems*, pp. 334-345.

Song, J., Guo, C., Wang, Z., Zhang, Y., Yu, G., Pierson, J., 2015. HaoLap: A Hadoop based OLAP system for big data. In *Journal of Systems and Software*, 102, pp. 167-181.

Sumbaly, R., Kreps, J., Shah, S., 2013. The "Big Data" Ecosystem at LinkedIn. In *ACM SIGMOD International Conference on Management of Data*, pp. 1125-1134.

Tardio, R., Mate, A., Trujillo, J., 2015. An Iterative Methodology for Big Data Management, Analysis and Visualization. In *BIG DATA'15 Proceedings of the 2015 IEEE International Conference on Big Data*, pp. 545-550.

Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E., 2015. Data wrangling: The challenging journey from the wild to the lake. In *Conference on Innovative Data Systems Research*.

Thakur, G., Gosain, A., 2011. DWEVOLVE: a Requirement Based Framework for Data Warehouse Evolution. In *ACM SIGSOFT Software Engineering Notes*, 36(6), pp. 1-8.

Thenmozhi, M., Vivekanandan, K., 2014. An Ontological Approach to Handle Multidimensional Schema Evolution for Data Warehouse. In *International Journal of Database Management Systems*, 6(3), pp. 33-52.

Thusoo, A., Sarma, J.S.., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R., 2010. Hive - a petabyte scale data warehouse using Hadoop. In *International Conference on Data Engineering*, pp. 996–1005.

White, T., 2009. *Hadoop: The Definitive Guide*, first ed., O'Reilly Media, Inc.

Wojciechowski, A., 2018. ETL workflow reparation by means of case-based reasoning. In *Information Systems Frontiers*, 20(1), pp.21-43.

Wu, L., Sumbaly, R., Riccomini, C., Koo, G., Kim, H.J., Kreps, J., Shah, S., 2012. Avatara: OLAP for Web-scale Analytics Products. In *VLDB Endowment*, 5(12), pp. 1874-1877.

Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I., 2010. Spark: Cluster Computing with Working Sets. In *HotCloud'10, 2nd USENIX*