International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.14, No.1 (Oct-23)

http://dx.doi.org/10.12785/ijcds/140195

Towards a Scalable and Efficient ETL

El Yazid Gueddoudj[1*] **and Azeddine Chikh**[1]

[1]*Computer Science Department, Faculty of Science, University of Tlemcen, Tlemcen 13000, Algeria*
[1]*Laboratoire LRIT, University of Tlemcen*

Received 23 Dec. 2022, Revised 28 May. 2023, Accepted 15 Sep. 2023, Published 1 Oct. 2023

Abstract: Extract, transform, and load (ETL) processes are crucial for building repositories of data from a variety of self-contained sources. Despite their complexity and cost, ETL processes have demonstrated some maturity for traditional, XML, and graph data sources. However the main challenge for ETL processes is double: (1) they do not scale when brought down to managing large and highly varied data sources, involving web-data. (2) the deployment of the target data warehouse in a polystore. The paper reviews various research efforts along this line of research. The paper then proposes a conceptual modeling of these processes using BPMN (Business Process Modeling Notation). These processes are automatically converted to scripts to be implemented within Spark framework. The solution is packaged according a new distributed architecture (Open ETL) that supports both batch and stream processing. To make our new approach more concrete and evaluable, a real case study using the LUBM benchmark, which involves heterogeneous data sources is considered.

Keywords: ETL, Scalability, Data Warehouse, Spark, Design, BPMN, Polystore.

1. INTRODUCTION

Today, all companies have data, which can be in any form: csv files, flat files, XML, relational or graph databases, etc. These data are captured, aggregated, cleaned, and loaded into a dimensional data warehouse that is denormalized. All of the processes related to the preparation of data for future analysis are called ETL (Extract-Transform-Load), which is widely used in business intelligence projects. Since the volume of data is rapidly increasing in enterprise systems, the ETL processes takes more time. Furthermore, the appearance of new storage platforms such as the polystore, has led to a heterogeneous ecosystem that has become difficult to manage and optimize. Accordingly, new large-scale data processing systems (e.g., Spark [1] or MapReduce (MR) [2]) have emerged. One of the difficulties of building a data warehouse application is managing the heterogeneity of information sources and storage platforms, because of the complex the design of ETL processes. Therefore, this research work aims at facilitating and optimizing the design and implementation of ETL processes.

A considerable amount of literature has been published ETL processes modeling. [3], propose a logical modeling for ETL processes using extended relational algebra with update operations. ETL tasks are expressed in XML. Moreover, [4] separates four types of models: ontology, UML, graph, and BPMN. A variety of approaches have been discussed to optimize ETL processes. In [5] the authors propose a conceptual model entity mapping diagram (EMD) in order to implement the ETL processes. The authors in [6], have developed an ETL based framework with deployment on the cloud platform. In [7] the author proposes a cloud-based architecture for a big data ETL. They use a general cluster-size optimization algorithm. In [8], the authors propose an ETL model in which applying the CDC (Change Data Capture) method in the Spark framework can reduce the amount of data in the ETL process. An ETL architecture for processing IoT streams with three components (the data collector, the ETL streaming engine, and the OLAP component) has been proposed by [9]. When it comes to storing and analyzing heterogeneous data, traditional data warehouses fail because they rely on RDBMS [10]. Few works that have addressed the association of the ETL with the polystore. In [9], the authors implemented the new Streaming ETL architecture for use in a relational context, they considered an ETL deployment on Intel's BigDAWG polystore. An ETL as a Service (ETLaaS) and Polystore as a Service (PaaS) deployment solution has been proposed by [11].

We argue that for modern generation data warehouses, in which we must take into account the problems of the heterogeneity of storage platforms, as well as the variety

E-mail: yazid.gueddoudj@univ-tlemcen.dz, azeddine.chikh@univ-tlemcen.dz*

and volume of data sources, linked to the VS of big data, a new ETL architecture should be designed. This paper aims to build a new architecture for robust, flexible, and cost-effective Spark-based ETL to optimize and make more efficient ETL processes by scaling them.

There is a substantial amount of published literature on models and architectures for optimizing ETL processes, but these models are still incomplete and deal with specific areas. Hence we propose in this paper :

- a new approach for designing ETL processes, which is based on Business Process Modeling Notation (BPMN). The former is a standard that provides a conceptual and implementation-independent specification of these processes, which is converted to executable specifications for ETL tools. This approach minimizes the development efforts because of the easiest detection of inconsistencies between the different elements of ETL processes. This helps in turn the designer estimating the development feasibility.

- a new solution for implementing robust and optimized ETL processes using the Spark framework. The former gives the possibility to extract and collect data from all types of sources, databases (RDBMS, NoSQL, Graph RDF); flat files; big data technologies (Hadoop, Spark); data generated by sensors, etc. This solution offers a deployment of the data warehouse on the Spark SQL polystore [12], depending on the target either the graph using Neo4j DBMS or the relational using Oracle DBMS.

From what we know, this work is the fist to describe how to conceptualize and optimize ETL processes using a set of BPMN notations for ETL operators capable of modeling ETL activities, as well as how to transform models into Scala scripts that are automatically converted into scripts to be implemented in the Spark framework.

The remainder of this paper is organized as follows. In section 2, we present a literature review on modeling and optimizing ETL processes. Our main contribution is presented in section 3, which is the main section of the paper. Section 4 a case study. Section 5 presents the experimental and the results. Section 6 concludes the paper and suggests future directions for this research.

2. RELATED WORK

We discuss in this section the state of the art on ETL processes design and optimization, which we classified in two categories : (1) conceptual modeling and (2) optimization.

Conceptual modeling works. Asma Dhaouadi et al. [13], provide an overview of pertinent studies concerning the modeling of data warehousing methods, encompassing traditional ETL processes as well as contemporary ELT design approaches. This work identifies the key obstacles and concerns associated with the development of Big Data warehousing systems. Some works have considered ontologies as resources intended to facilitate and automate the design of the ETL process. Authors in [14] detail the formal concepts of a conceptual model for the ETL process. The required design standards and methodologies, however, were not covered. Using a set of steps, the authors of [15] have established an approach to build a conceptual model for ETL that may be customized. The issue of semantic and structural heterogeneity is addressed in [16]. The authors uses an ontology to automatically derive ETL transformations. In [17],the authors address the issue of selecting appropriate transformations and defining inter-schema mappings for both structured and semi-structured data. In order to identify pertinent data sources and specify the conceptual ETL processes for integrating the data in the target warehouse, an ETL algorithm is proposed, this approach is an informal description of the ETL process. A semantic ETL framework is proposed by the authors in [18], which makes use of semantic technologies to combine and make available data from many sources as linked open data. To offer more abstraction, the conceptual modeling of ETL processus using BPMN notation is suggested by the authors. By using Business Process Modeling Notation, the authors of [19] propose a conceptual ETL process modeling. The issue of updating slowly changing dimensions (SCD) with dependencies, or the situation where the update of one SCD table has an effect on the connected SCD tables, is addressed in this work. In the context of multi-model data warehousing and and to improve the ETL performance, various options arise for representing dimensions and facts logically. Sandro Bimonte et al. [20], propose guidelines for multimodel multidimensional design in MMDBMS-based data warehouses, validated through intra-model and inter-model comparisons, and illustrated with a case study. In [21], the authors consider the variety of data sources and the variety of data warehouse storage platforms and their impact on the efficiency and performance of the ETL process. A new approach to the design of ETL processes using models and meta-models is described. Furthermore, a new approach to modeling the ETL process using the modeling language (SysML) is proposed by authors in [22]. A simulation of an expected case study of e-commerce is presented. In [23], the authors describe a semantic ETL process, where the ten existing operators are specified on graphs from the data sources to the data warehouse.

Optimization works. To improve the efficiency of ETL processes, the authors in [24] and [25], proposed a state-space based search method that models an ETL process as a state, to achieve the goal of optimizing execution. Algorithms for minimizing the cost of running the ETL process are presented. They have been implemented in C++ and tested for variations in time and the volume of visited states. The authors in [26] propose an optimization framework using partitioning and parallelization of ETL processes to minimize the time and resources required for data streams ETL as part of their work on the parallelism

of ETL streams. For data integration, the framework is built on the open source ETL tool Talend Open Studio. An on-demand incremental ETL solution based on the query-fetch-transform-load paradigm is presented by the authors in a recent paper [27]. The authors propose that the extraction phase of the ETL process be optimized along with data reuse. The authors of [28], explore how MapReduce systems and database systems differ in their architectural choices, they concluded that the MR use case is similar to an ETL system. The authors in [29], present the ETLMR framework, which uses MapReduce to achieve scalability. Therefore, it easily creates dimensional ETL processes based on MapReduce to load data into the data warehouse. The user can implement parallel ETL programs with a few lines of code declaring target tables and transformation functions. A cloud-based ETL solution, as well as a new Spark-based ETL framework, are presented in [6]. The authors in [30] propose a new approach that dynamically combines the ETL process and the selection of materialized views, within the framework of a near real-time data warehouse design and considers semantic data sources. The authors present a view selection methodology with three components: the main memory and cache management; the dynamic selection of MV (materialized views); and a stream ETL process: starts by extracting the instances from the RDF data sources and saving them temporarily in the buffer of inputs. In [31], the authors present a new distributed ETL architecture, Striim developed to collect datasets from different sources and transform them to aggregate without considering the same or different timestamps. The ETL engine runs in a scalable and fault-tolerant cluster of compute nodes. It extracts real-time data from sources, transforms it, and produces result events that are loaded or broadcast into targets. ETL processes are created by authors in [32] by writing Python code. By utilizing built-in functionality and data access for fact and dimension tables, developers could handle data efficiently. PostgreSQL, MySQL, and Oracle databases are all supported. There is no cost model provided by the authors' solutions to quantify the performance gain. Mehmood and Anees in [33], propose a real time ETL architecture for unstructured big data during the transformation phase, focusing on accelerating stream disk joins and mitigating disk overhead and data loss. Finally in order to improve the performance of the ETL process in the era of big data and by pre-processing the data beforehand, Monika Patel and Dhiren B. [34], propose an ETL framework for data warehousing utilizing and leveraging NoSQL databases technologies.

3. OUR CONTRIBUTION AN OPEN ETL APPROACH

In this section, we present the main features and principles of our Open ETL approach shown in figure 1. First, we present how to create a BPMN ETL design. Second, how to adapted the traditional ETL process in order to distribute it and allow its execution in parallel on a Spark cluster. We highlight the importance of data partitioning, in a distributed environment and then present the types of data partitioning provided in our approach. Our approach to optimizing ETL processes is to parallelize ETL processes using the Spark framework, which supports in-memory data sharing between multiple tasks. We translate the BPMN model directly into scripts written in Scala language, following the Spark execution model illustrated in figure 2. Scala scripts are automatically parallelized and run on a three-machine cluster (one Namenode and two Datanodes). The execution system takes care of the details of the partitioning of the input data; the planning of the execution of the program on a set of machines, the management of; and inter-machine communication. Since traditional data warehouses are not optimized enough to store huge and diverse amounts of data [35], in our approach we use a new data warehousing architecture such as polystore.

A. The New Architecture

The architecture of the proposed approach is illustrated in figure 1. The proposed approach consists in: Designing an ETL process coded in Scala scripts involving heterogeneous data sources in order to ensure scalability and reduce time latency between a data warehouse and its sources. The warehouse is deployed on the Spark SQL polystore. The proposed method consists in five steps:

1) Extraction and transfer of stream data: it consists in is an update from the data sources to the distributed file system HDFS (Hadoop Distributed File System). We used the capture technique that we coded in Scala;
2) Modeling: it consists in producing a "pivot" UML model from the models of the different data sources;
3) Mapping : it consists in transforming each BPMN object into its equivalent in the Scala script;
4) Transformation : it consists in making the appropriate transformations with Scala scripts using both the traditional ETL operators and the ETL operators for the management of graph representations defined in [23];
5) ETL deployment on the Spark SQL polystore.

The Spark ETL engine. Apache Spark is a cluster computing platform with built-in libraries and simple Python, Java, Scala, and SQL APIs that is intended to be quick and all-purpose. The MapReduce concept is extended by Spark to effectively handle additional computing types, such as interactive queries and streaming processing. For complex applications operating on disk [36], Spark is more effective than MapReduce because it allows calculations to be performed in memory. Spark's Resilient Distributed Dataset serves as its foundation (RDD).

Parallel Processing in Spark. Spark applications run as independent processes that reside on clusters and are coordinated by SparkContext in the main program. The first step in running a Spark program is to submit the job to a cluster using spark-submit. Sparkcontext routes the program to the modules, such as the Cluster Master Node,

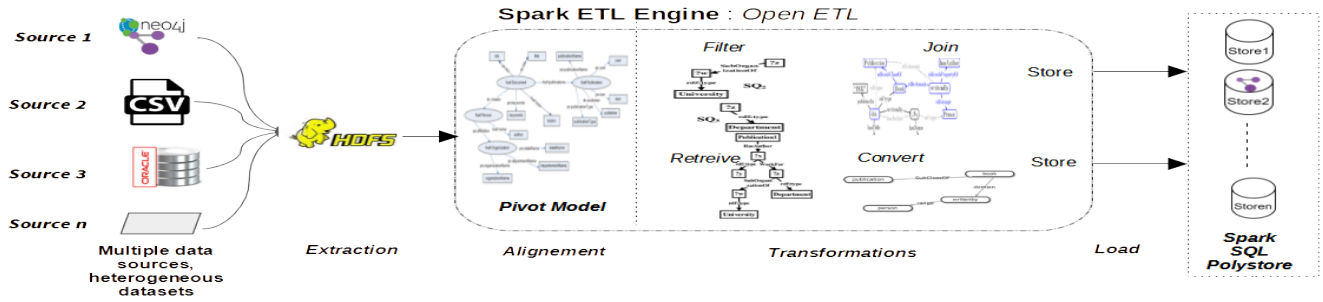

Figure 1. General Architecture of our Approach.

as well as the RDDs are created by these Sparkcontext driver programs. The program is then transmitted to the cluster master node. Each cluster has a master node that does all the necessary processing. It then transmits the program to the worker nodes. The working node is the problem solver. Master nodes contain executors that run with the Sparkcontext driver.

RDD vs ETL process. A great deal of correspondence and similarity exists between the ETL and RDD processes. RDDs are immutable, distributed collections of objects of any type. As the name suggests, these are resilient (fault-tolerant) records of data that reside on multiple nodes. Intuitively, an ETL process can be thought of as a directed acyclic graph (DAG), with the activities and record sets being the nodes of the graph and the input-output relations between the nodes being the edges of the graph [37]. In RDD, when the computation set is created, forming a DAG, it does not perform any execution, but it prepares for execution at the end. After the extraction, the ETL process performs two types of transformations on the data: standardization of data (cleaning, filtering, conversion, ...) and allows the second to merge, aggregate. Thus, the ETL process fits well with Spark's RDD execution model.

Dataset and DataFrame. A dataset is a distributed collection of data. A DataFrame is a dataset organized in columns that bear names, like the tables of a database. It is possible to apply to the dataFrame actions, which produce values; and transformations, which produce new dataFrames or datasets, as well as other functions.

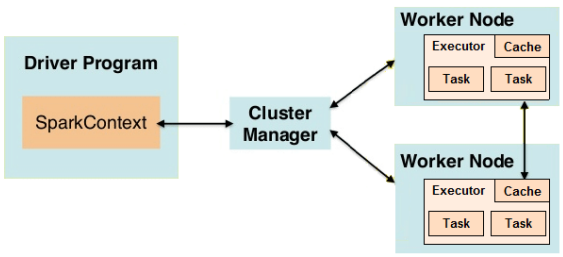

Figure 2. Spark Execution Model. [38]

A Pivot model. Integrating heterogeneous data with heterogeneous data models into data warehouse applications leads to various complexities, such as semantic and structural heterogeneity, data redundancy, and the maintenance of different data models. To minimize this complexity, we adapt the pivot model introduced in [39]. It provides a generic illustration of various data source formalisms; in our case, we chose alignment to the RDF data model. With n sources of data involved in the integration, each source has its formalism. One has to perform $(n*(n-1)/2)$ different mappings. We can, however, reduce the number of mappings by using a pivot model.

B. ETL process meta-model

This section's goal is to provide a meta-model of the ETL process that supports generic modeling of the activities in the ETL environment. An ETL process can be considered a special type of business process. An ETL process is a special type of business process, but there is no standard model to define this process. As with any process, the ETL environment must first be modeled before any of its components can be used. A significant number of research focused mostly on modeling ETL activities to produce the ETL process [4], [40], [14] and [41].

Definition. An ETL process is made up of many activities and transitions between them. A transition determines the order in which activities are executed to provide a data flow from sources to destination stores, and an activity corresponds to one ETL scenario. Elements from data sources, stores, and ETL operators are all used in an activity to define expressions that describe the semantics of the data pushed to the target schema. Each activity is either simple, impossible to break down, or complex. In order to contribute to the generalization of the entire ETL environment, we provide a meta model of the ETL process that allows a generic modeling of the ETL environment's activities. In (figures 3 and 4), we describe the constructions composing the proposed ETL palette. These constructions are grouped into four categories: operators [17], activity objects, and workflow objects. For the other objects (artifacts, connections, control, and flow), we use the constructions proposed by the Camunda modeler tool [42].

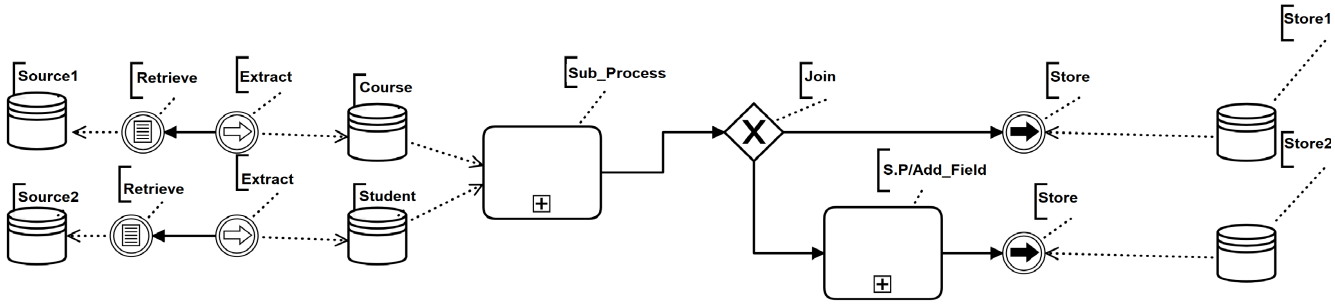

Figure 3. ETL Activity.

C. Model To Script Template

To translate the ETL conceptual model, we do the work of designers, similar to what happens when translating conceptual models, such as the Entity-Association model, into logical models. Additionally, the designer should break down each high-level activity into a detailed sub-process and repeat the process until the implementable ETL operators are reached. Then comes the mapping, which consists in transforming each BPMN object into its equivalent in the Scala script.

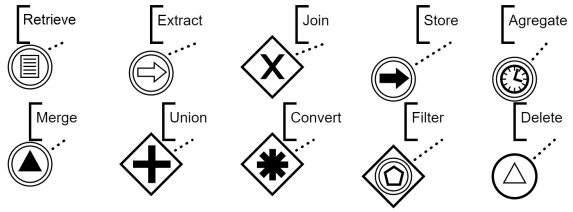

Figure 4. ETL Operators.

D. Polystore Deployment

[43] divides existing solutions for multi-model data management into four groups: federated systems, polyglot systems, multi-store systems, and polystore systems. A polystore system is a database system with many heterogeneous data stores and various query interfaces, according to [44]. In our solution, we choose to deploy the target data warehouse on the Spark SQL hybrid polystore whose architecture is shown in figure 5, Spark SQL polystore architecture, based on [12], p.29. This architecture runs as a library on top of Spark. According to the target, (i) in a vertical representation using the DBMS Neo4j, in which one can represent instances and graphs, and (ii) in a relational representation using the Oracle DBMS.

We applied our Scala script-based ETL program to populate the target data warehouse schema. We assume that with n data sources we can deploy the final data warehouse according to the requirements announced beforehand by the designer of the warehouse. Spark SQL is a hybrid polystore system with loosely coupled external data sources and tightly coupled Spark/HDFS. On top of Spark, it functions as a library. The Spark Java interface provides direct access to the Spark engine for the query processor, while using wrappers to access external data sources via the common Spark SQL interface (such as a relational DBMS or a Neo4j store).

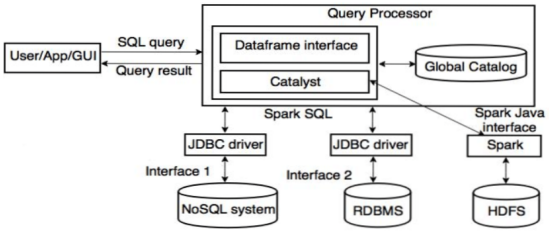

Figure 5. Spark SQL polystore architecture. [45]

4. CASE STUDY

We illustrate an example of scenario through a data warehouse. The training department of the Ministry of Higher Education and Research wants to build a data warehouse that consists in collecting decisional information on students performance, namely their performance.

A. Data-Sets

The schema of the data warehouse is shown in figure 6, taken from the LUBM benchmark (http://swat.cse.lehigh.edu/projects/lubm/) (figure 7) related to the university's domain. along with the overall process of loading data from sources (we present the example of an RDF file, figure 8) to the target data warehouse.

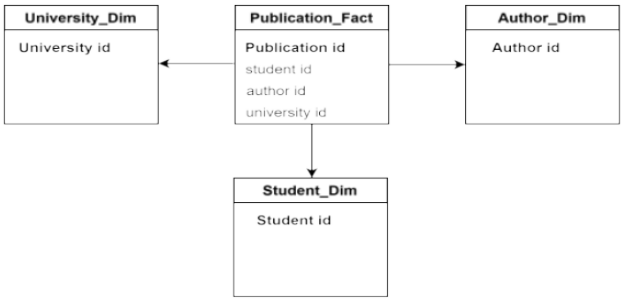

Figure 6. Data Warehouse Schema Design.

A tool named UBA is provided by LUBM to generate data on the Univ-Bench ontology. We used these datasets to simulate graphs based on external data sources. We also considered CSV data sources and a relational database as internal sources. (i) Source 1 two Neo4j graph database populated using the Lehigh University Benchmark (LUBM) ontology; (ii) Source 2 is a CSV file; and (iii) Source 3 is an Oracle database. In our case, the destination tables are initially empty and then filled with new data. To do this, a pivot model (RDF schema) is designed. This data model provides a basis for integration.

B. Measures

We evaluate the performance of the proposed system through a set of experiments by considering three criteria: (i) the scalability, (ii) the response time of our approach compared to an ETL tool PDI (Pentaho data integration tool) [46], and (iii) the time spent loading the changes that affected the data sources.

C. ETL Process

ETL script code imports data from an RDF file to multiple nodes with Spark. Spark changes the transformations from RDD to DAG (Directed Acyclic Graph) and starts execution. DAG converts a logical execution plan into a physical execution plan. When an activity is invoked, the DAG is submitted to the DAG scheduler. It divides operations into steps, and each step consists in a unit of work called a task. These tasks are then transmitted to the task manager via a cluster manager, which makes it possible to launch the application on different machines. Figure 8 depicts the entire ETL process applied to the RDF file.

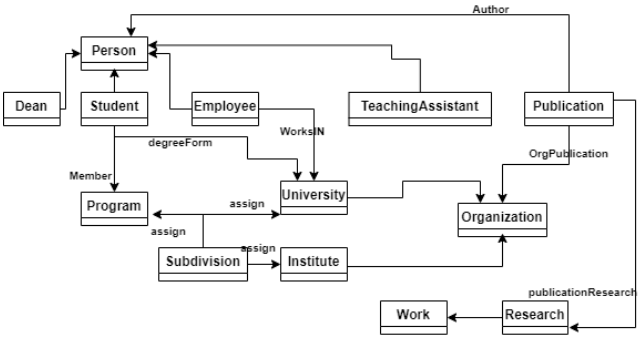

Figure 7. Schema of the LUBM ontology.

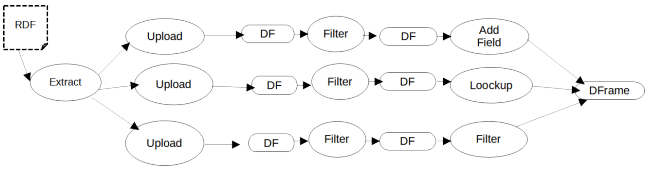

Figure 8. Partitioning and distribution in Spark.

After applying our Open ETL algorithm, the ETL process in BPMN is shown in Figure 9, the resulting data warehouse has two stores: a Neo4j graph database and an Oracle database. The processing is done in the Spark framework from a distribution perspective.

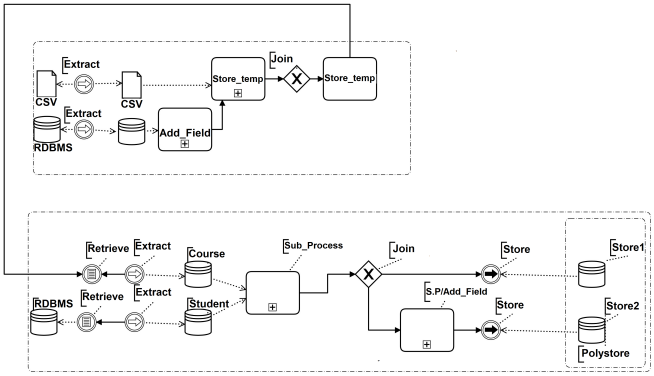

Figure 9. The Open ETL Process.

5. EXPERIMENT AND RESULTS

This section presents the evaluation of the proposed system, which is based on the aforementioned case study. We conducted three separate experiments. The initial experiment examined the scalability of the proposed solution by varying the size of the data source. The second experiment focused on evaluating and comparing the response time of the proposed system with the PDI tool. The third experiment focused on evaluating the loading time of the ETL process after a change in the sources.

Our experiments were conducted on a PC with an Intel(R) Core(TM) i5-8365U processor (1.9 GHz, 1.6 GHz), 8 GB of main memory on a 256 GB SSD running Windows 10x64, Oracle DBMS, Neo4J desktop database, Apache Spark 3.0, and Scala 2.12 were required to run the tests. All of the chosen software tools were installed on the local machine for evaluation purposes.

In Experiment 1. In this experiment, we considered the scalability of our approach by varying the size of the data sources and, therefore, the number of tasks that run in parallel on the cluster to process the data. The translation of the BPMN conceptual model of ETL processes into Scala scripts allows for the populating of the target schema of the data warehouse. We measure the time spent in integrating heterogeneous data sources of different sizes. Figure 10 illustrates the results obtained. We can remark that increasing the size of the sources improves the processing time.

In Experiment 2. In this experiment, we aim to show that modeling the ETL process with BPMN and translating it into scripts coded in Scala and running on Spark results in more efficient processes than those resulting from translating BPMN into ETL tools. We evaluate the response time of our proposal compared to the PDI tool. The results are shown in figure 11. We note that our approach considerably increases the response time of the ETL process.

In Experiment 3. In this experiment we evaluate the loading time of the ETL process after a change in the

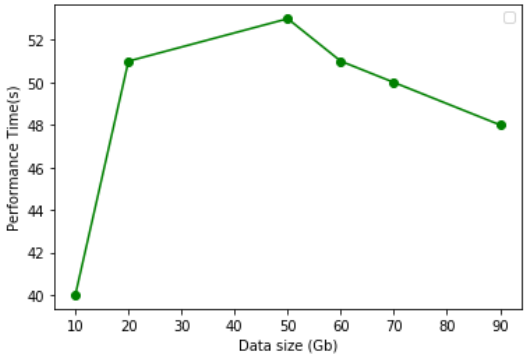

Figure 10. Scalability of the proposed approach.

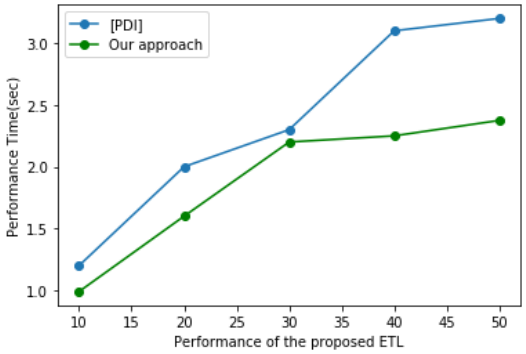

Figure 11. Performance of the proposed ETL.

sources. Figure 12 depicts our discovery that the loading time increases after an update. The loading via our approach is superior to that implemented on the PDI tool.

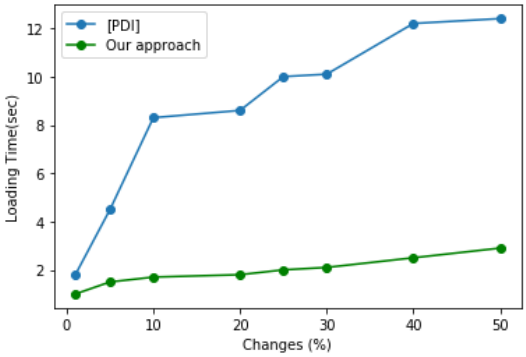

Figure 12. Load Time Comparison.

6. CONCLUSIONS AND FUTURE WORK

This paper introduced a new method that includes both methods of modeling and optimization of the ETL processes, respecting the importance of the issue and the challenges in this area. We presented a new approach for optimizing ETL processes through a new distributed architecture (Open ETL) that supports both batch and stream processing. We presented a new approach to design ETL processes using a set of notations in BPMN for ETL operators capable of modeling ETL activities as well as model transformations into Scala scripts which are automatically converted into scripts to be implemented in the Spark framework. A hybrid polystore deployment solution based on Spark SQL is given. We validated the concrete steps in using the proposed Open ETL approach to load data into a data warehouse schema. The important results are related to the performance time of the proposed system compared to other ETL alternatives. In addition, we have also presented the scalability of the proposed method on large data sets. Thus, future work may focus to design a unified code-based ETL framework involving time-series data management support. Also, in our future research we plan comparing the performance of our approach with other ETL alternatives.

REFERENCES

[1] A. Inc. (2018) Apache spark. [Online]. Available: http://spark.apache.org/

[2] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] J. Awiti, "Algorithms and architecture for managing evolving etl workflows," in *European Conference on Advances in Databases and Information Systems*. Springer, 2019, pp. 539–545.

[4] S. M. F. Ali and R. Wrembel, "From conceptual design to performance optimization of etl workflows: current state of research and open problems," *The VLDB Journal*, vol. 26, no. 6, pp. 777–801, 2017.

[5] S. H. A. El-Sappagh, A. M. A. Hendawi, and A. H. El Bastawissy, "A proposed model for data warehouse etl processes," *Journal of King Saud University-Computer and Information Sciences*, vol. 23, no. 2, pp. 91–104, 2011.

[6] N. Biswas and K. C. Mondal, "Integration of etl in cloud using spark for streaming data," in *International Conference on Emerging Applications of Information Technology*. Springer, 2021, pp. 172–182.

[7] E. Zdravevski, P. Lameski, A. Dimitrievski, M. Grzegorowski, and C. Apanowicz, "Cluster-size optimization within a cloud-based etl framework for big data," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 3754–3763.

[8] I. P. M. Atmaja, A. Saptawijaya, S. Aminah *et al.*, "Implementation of change data capture in etl process for data warehouse using hdfs and apache spark," in *2017 International Workshop on Big Data and Information Security (IWBIS)*. IEEE, 2017, pp. 49–55.

[9] J. Meehan, C. Aslantas, S. Zdonik, N. Tatbul, and J. Du, "Data ingestion for the connected world."

[10] S. Bimonte, E. Gallinucci, P. Marcel, and S. Rizzi, "Data variety, come as you are in multi-model data warehouses," *Information Systems*, vol. 104, p. 101734, 2022.

[11] N. Berkani and L. Bellatreche, "Streaming etl in polystore era," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2018, pp. 560–574.

[12] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, 2015, pp. 1383–1394.

[13] A. Dhaouadi, K. Bousselmi, M. M. Gammoudi, S. Monnet, and S. Hammoudi, "Data warehousing process modeling from classical approaches to new trends: Main features and comparisons," *Data*, vol. 7, no. 8, p. 113, 2022.

[14] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual modeling for etl processes," in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, 2002, pp. 14–21.

[15] P. Vassiliadis, A. Simitsis, P. Georgantas, and M. Terrovitis, "A methodology for the conceptual modeling of etl processes," in *CAiSE Workshop on Decision Systems Engineering*, 2003.

[16] D. Skoutas and A. Simitsis, "Designing etl processes using semantic web technologies," in *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, 2006, pp. 67–74.

[17] ——, "Ontology-based conceptual design of etl processes for both structured and semi-structured data," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 3, no. 4, pp. 1–24, 2007.

[18] N. Berkani, L. Bellatreche, and S. Khouri, "Towards a conceptualization of etl and physical storage of semantic data warehouses as a service," *Cluster computing*, vol. 16, no. 4, pp. 915–931, 2013.

[19] J. Awiti, A. A. Vaisman, and E. Zimányi, "Design and implementation of etl processes using bpmn and relational algebra," *Data & Knowledge Engineering*, vol. 129, p. 101837, 2020.

[20] S. Bimonte, E. Gallinucci, P. Marcel, and S. Rizzi, "Logical design of multi-model data warehouses," *Knowledge and Information Systems*, vol. 65, no. 3, pp. 1067–1103, 2023.

[21] N. Berkani, L. Bellatreche, and L. Guittet, "Etl processes in the era of variety," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIX*. Springer, 2018, pp. 98–129.

[22] N. Biswas, S. Chattapadhyay, G. Mahapatra, S. Chatterjee, and K. C. Mondal, "A new approach for conceptual extraction-transformation-loading process modeling," *International Journal of Ambient Computing and Intelligence (IJACI)*, vol. 10, no. 1, pp. 30–45, 2019.

[23] N. Berkani, L. Bellatreche, and B. Benatallah, "A value-added approach to design bi applications," in *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2016, pp. 361–375.

[24] A. Simitsis, P. Vassiliadis, and T. Sellis, "State-space optimization of etl workflows," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 10, pp. 1404–1419, 2005.

[25] ——, "Optimizing etl processes in data warehouses," in *21st International Conference on Data Engineering (ICDE'05)*. Ieee, 2005, pp. 564–575.

[26] X. Liu and N. Iftikhar, "An etl optimization framework using partitioning and parallelization," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, pp. 1015–1022.

[27] L. Baldacci, M. Golfarelli, S. Graziani, and S. Rizzi, "Qetl: An approach to on-demand etl from non-owned data sources," *Data & Knowledge Engineering*, vol. 112, pp. 17–37, 2017.

[28] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "Mapreduce and parallel dbmss: friends or foes?" *Communications of the ACM*, vol. 53, no. 1, pp. 64–71, 2010.

[29] X. Liu, C. Thomsen, and T. B. Pedersen, "Mapreduce-based dimensional etl made easy," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1882–1885, 2012.

[30] N. Berkani, L. Bellatreche, and C. Ordonez, "Etl-aware materialized view selection in semantic data stream warehouses," in *2018 12th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2018, pp. 1–11.

[31] A. Pareek, B. Khaladkar, R. Sen, B. Onat, V. Nadimpalli, and M. Lakshminarayanan, "Real-time etl in striim," in *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics*, 2018, pp. 1–10.

[32] C. Thomsen and T. B. Pedersen, "Easy and effective parallel programmable etl," in *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*, 2011, pp. 37–44.

[33] E. Mehmood and T. Anees, "Distributed real-time etl architecture for unstructured big data," *Knowledge and Information Systems*, vol. 64, no. 12, pp. 3419–3445, 2022.

[34] M. Patel and D. B. Patel, "Data warehouse modernization using document-oriented etl framework for real time analytics," in *Rising Threats in Expert Applications and Solutions: Proceedings of FICR-TEAS 2022*. Springer, 2022, pp. 33–41.

[35] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik, "The bigdawg polystore system," *ACM Sigmod Record*, vol. 44, no. 2, pp. 11–16, 2015.

[36] C. R. Valêncio, M. H. Marioto, G. F. D. Zafalon, J. M. Machado, and J. C. Momente, "Real time delta extraction based on triggers to support data warehousing," in *2013 International Conference on Parallel and Distributed Computing, Applications and Technologies*. Ieee, 2013, pp. 293–297.

[37] P. Vassiliadis and A. Simitsis, "Extraction, transformation, and loading." *Encyclopedia of Database Systems*, vol. 10, 2009.

[38] H. Nazeer, W. Iqbal, F. Bokhari, F. Bukhari, and S. U. R. Baig, "Real-time text analytics pipeline using open-source big data tools," *arXiv preprint arXiv:1712.04344*, 2017.

[39] I. Boukhari, L. Bellatreche, and S. Jean, "An ontological pivot model to interoperate heterogeneous user requirements," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer, 2012, pp. 344–358.

[40] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis, and S. Skiadopoulos, "A generic and customizable framework for the design of etl scenarios," *Information Systems*, vol. 30, no. 7, pp. 492–525, 2005.

[41] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Modeling etl activities as graphs." in *DMDW*, vol. 58, 2002, pp. 52–61.

[42] C. Inc. (2020) Camunda modeler. [Online]. Available: https://camunda.com/platform/modeler/

[43] R. Tan, R. Chirkova, V. Gadepally, and T. G. Mattson, "Enabling query processing across heterogeneous data models: A survey," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 3211–3220.

[44] P. Chen, V. Gadepally, and M. Stonebraker, "The bigdawg monitoring framework," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2016, pp. 1–6.

[45] C. Bondiombouy and P. Valduriez, "Query processing in multistore systems: an overview," *International Journal of Cloud Computing*, vol. 5, no. 4, pp. 309–346, 2016.

[46] H. Inc. (2018) Pentaho data integration. [Online]. Available: https://help.hitachivantara.com/Documentation/Pentaho/8.3/Products/Pentaho_Data_Integration

El Yazid Gueddoudj is PhD student in Computer Science, attached to LRIT Laboratory, University of Tlemcen, Algeria. He holds a Master in Intelligent models and decision from Tlemcen University, Engineering degree Computer Science from University of Setif, Algeria. His current research interests include Data Integration Systems, Data Warehouse Design, Big data and Ontology-based Modeling. He had more than 16 years of multi-functional experience achievements, especially as a teacher, computer engineer, product data specialist, project coordinator and headmaster in administration.

Pr. Azeddine Chikh is Professor in Computer Science and Director of Research Laboratory (LRIT) at University of Tlemcen, Algeria. He is also head of PHD program committee and the Mathematics-Informatics Domain at the same university. He worked before as Associate Professor at King Saud University-Saudi Arabia. He holds a PhD in Information Systems from National Institute of Computer Science in Algeria and Paul Sabatier University in France, a Master in e-learning from Switzerland, and a graduate certificate in Systems Engineering from the University of Missoury Rolla, USA. His research interests include Software Requirements Engineering; Communities of Practice; and Semantic Web.