AN INNOVATIVE METHOD TO EXTRACT DATA IN A REAL-TIME DATA WAREHOUSING ENVIRONMENT

Flavio de Assis Vilela[1] and Ricardo Rodrigues Ciferri[2]

[1]Departament of Computing, Federal Institute of Goiás, Jataí, GO
[2]Departament of Computing, Federal University of São Carlos,
São Carlos, SP

ABSTRACT

ETL (Extract, Transform, and Load) is an essential process required to perform data extraction in knowledge discovery in databases and in data warehousing environments. The ETL process aims to gather data that is available from operational sources, process and store them into an integrated data repository. Also, the ETL process can be performed in a real-time data warehousing environment and store data into a data warehouse. This paper presents a new and innovative method named Data Extraction Magnet (DEM) to perform the extraction phase of ETL process in a real-time data warehousing environment based on non-intrusive, tag and parallelism concepts. DEM has been validated on a dairy farming domain using synthetic data. The results showed a great performance gain in comparison to the traditional trigger technique and the attendance of real-time requirements.

KEYWORDS

ETL, real-time, data warehousing, data extraction.

1. INTRODUCTION

In recent years, the collect of data that support the decision-making process has become fundamental for different applications domains, such as health care systems, road control systems, retail systems and smart farm systems [1], [2], [3], [4]. These data are located in operational sources (OLTP systems), that is, in the placement where occurs transactions issued by business applications. Furthermore, the operational data are available in different formats from information providers (data sources) that are autonomous, heterogeneous, and distributed. Thus, there are a lot of ways to store the data of interest for the decision-making process, such as in files stored in computer desktops, in relational databases using centralized or distributed servers, in mobile applications, in semi-structured and non-structured sources, in text files, XML or JSON files, in spreadsheets and other formats, and storage medias [5], [6].

The decision-making process is performed based on the data of interest gathered from the operation sources (data sources). Therefore, to provide value for the operational data and effectively aid the decision-making process, it should be used the ETL (Extract, Transform, and Load) process. This process is responsible for dealing with the operational data and performing the extraction, cleaning, processing, and integration of data. Thus, the data of interest available in the operational environment are sent to a data warehousing environment and stored into a multidimensional, homogeneous, and integrated database called data warehouse [7], [8], [5].

David C. Wyld et al. (Eds): CSE, AI & FL, NLPTT - 2021
pp. 01-17, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.112401

In the traditional ETL process, the process for gathering data from the data sources frequently takes place in a specific and pre-defined period of time, in a loading time window or according to the organizations business rules. Generally, the frequency of updates is on a daily, weekly, monthly basis and occurs in off peak hours. Furthermore, it takes place in a predefined frequency that is suitable to the data requirements for the decision-making process in organizations [9], [5], [10]. When using the loading time window for gathering data, both operational and informational environments should be offline to perform the process. It means that while updating, the OLAP (on-line analytical processing) applications cannot access any up-to-date data. This fact narrows the ETL process to the highest time of the loading time window. Furthermore, it imposes challenges to the organizations that have branch offices around the world, which time zone prevent to defining the same loading time window period. According to Muddasir e Mohammed [11], by using the loading time window period is just suitable to organizations which allow this feature or that focus on long run goals, which data can be updated in a less frequency. So, the usage of loading time window period to update data is enough to the majority applications, as it had been widely studied and applied in OLAP applications with low cost when is used the incremental data updating approach [12].

On the other hand, from the evolution in the way that data are generated and handled for decision-making process, which are generated quickly, a lot of applications has been emerged in which needs gathering operational data from operational sources in a real-time, such as health care systems [13] and systems that handle data from sensors [14]. Moreover, some applications, such as digital agriculture systems [3], [15], [16] and road traffic systems [17], not always generate large volume of data, but they need to guarantee that the data will be available in real-time into data warehouse to making-decision process[12], [18].

The real-time requirement can be classified as hard real-time or soft real-time. We adopt the soft real-time, where the failure to meet the timing constraint requirements does not cause serious damages. So, performing all response time rigorously is not the only aspect to be considered. One example of the system that applies this concept is on-line transaction systems, which performs a lot of tasks in a real-time.

According to Sabtu et al. [19], the updating feature of traditional ETL process can negatively and critically affect the data analysis process for the applications that have this real-time requirement. This is because the results of OLAP queries that were obtained from data warehouse throughout the day can return inconsistent data in relation to the current organization situation. In other words, data warehouse is updated few times a day and the operational sources generate data continuously [20], [11]. So, this fact makes narrows to adopt the traditional ETL process in the context of data warehousing real-time requirements.

This paper proposes a new innovative method named Data Extraction Magnet to solve the extraction phase of ETL process in real-time data warehousing environments. DEM method was designed to provide low response time, scalability, decoupling between ETL process, operational environment, and soft real-time requirements. Moreover, from all available techniques to traditional ETL process, DEM is a change of paradigm as a solution to perform extraction phase of ETL process in real-time data warehousing environments, since DEM is not intrusive to gather the data of interest in data sources. The experimental tests showed a great performance gain of DEM in comparison to the traditional trigger technique.

This paper is organized as follows: Section II presents the concepts of traditional, on-demand, near real-time and real-time approaches for performing ETL process. Section III summarizes the most important related work, Section IV presents the proposed Data Extraction Magnet (DEM), Section V discusses the performed experiments to validate the DEM and Section VI concludes the paper.

2. ETL

A data warehousing environment is the most important component of an informational environment, which supports data management and decision-making processes. Its main goal is to promote new decision markets, identify possible failures in the enterprise organizational process, plan new tasks, define analysis of tendency and other types of analysis [21]. The data warehousing environment is composed of architectures, algorithms, tools, and systems that make possible to obtain data from information providers to make available them in a dimensional, homogeneous, and integrated repository called data warehouse [21]. Furthermore, the data warehousing environment is characterized by dealing with historical data, that is, it makes use of the facts that already happened between ten years before or more to support the decision-making process. As a consequence of data accumulated during a large previous period of time, the data volume handled by the data warehousing environment is huge and increasing quickly, and it can easily reach Petabytes of storage [21], [8], [11].

ETL process is an inseparable part of the data warehousing environment, and it is composed and represented by a workflow of tasks. These tasks are logically split in three steps: 1) the first one is called Extraction (E). The main goal is to extract or capture the data of interest from data sources. This process is performed by using a wrapper or some techniques inherent to data storage systems, such as trigger or log files. In this step, it takes place the communication to operational sources, where the data are stored in the operation environment, each data source with its specific format and access path; 2) the next step is called Transformation (T), where is performed the data cleaning, data processing and data integration. In this step, the data is cleaned and, therefore, it is eliminated the conflicts of value, structural conflicts, and semantic conflicts for the data, such as synonyms, namesake, and inconsistent values. The cleaned and transformed values are stored in a temporally data source called data staging area (DSA), which standardizes the data format, which is independent of the source formats; 3) the third step is called Loading (L), which is responsible for load data to data warehouse from DSA [9], [22]. The result of the ETL process is to make the data of interest of the operational environment available to the decision-making process, that is, storing operational data into the data warehouse in an integrated and consistent way [9], [23].

The ETL process is based on data collected from data sources in the operational environment. These data can be generated in different ways, such as: 1) manually, through users interactions with OLAP systems, such as management control systems, social media and Web systems; 2) automatically, by systems, using batch processing; 3) automatically, through sensors, such as presence sensor, road sensors and sensors connected to people or animals body; 4) automatically, by means of satellite, which sending data constantly to your stations (i.e., climate data and vehicle traffic).

In addition, there are three main ETL approaches to update data from data sources to the data warehouse: 1) on-demand: the process of extracting and loading are performed, and operational data are stored directly into the data warehouse without applying transformation, cleaning, processing, or integration data processes. For each requested OLAP query, the transformation, cleaning, processing, and integration processes are performed on the fly and the query results are shown to the user. Thus, the main part of the ETL process takes place in posteriori, that is, at query processing moment. In this way, the drawback is the high response time to answer the OLAP query; 2) near real-time: the data are stored into the data warehouse in a lower response time than traditional way. In other words, there are organizations that, not necessarily, needs data in a real-time, but just in a lower response time. Thus, through this approach, the ETL process is performed at a higher frequency throughout the day. In this case, it is acceptable to have a certain latency, such as each hour or each minute. According to Muddasir and Mohammed [11], Kakish and Kraft [23], Langseth, J. [24], Sabry and Ali [8], there are three different techniques to be applied to the ETL process to reach the near real-time approach goals. These techniques are: *Direct trickle feed*, *Trickle and flip* and *External real-time data cache*. Basically, the most difference in each technique is just the way to handle the historical and new data. However, all techniques are performed using an intrusive strategy. An intrusive strategy needs to directly access the operational sources and also deal with the way to connect to the data sources and the heterogeneity of the operational sources.; 3) real-time: this approach stores the data into the data warehouse in a real-time manner. The data is selected from data sources and stored in the data warehouse immediately after some insert event occurs in the data source or in as low response time as possible. However, there is an implicit overhead in data exchange between the operational environment and data warehousing environment. It is due to hardware limitations, network protocol, limitations of transfer rate of data or whatever type of restrictions that act to increase the time to send data to the target data warehouse. [7], [18], [11], [9], [25].

The immediate solution to meet real-time is to increase the data warehouse data updating frequency, that is, the ETL process can be performed more than once a day and in each application idle time [20], [7], [9]. However, this solution can have three main drawbacks: 1) data reading from operational sources: the data sources from the operational environment are used to serve a group of users that performs transactions continuously. By using traditional ETL process, all available techniques are intrusive, that is, they need do access directly the operational sources to gather the data. So, it causes a query overloading every time that ETL process access the operational sources to extract data stored into these sources. This fact can cause a performance degradation of operational sources; 2) data writing into an informational environment: the data warehouse is used to serve a group of users that execute OLAP queries over the data. These queries are complex and involve a large data volume which in turn demand an excessive time to be performed. Thus, the data warehouse will be unavailable to be updated until the end of query processing; 3) updating time and data volume: when both environments are idle, the updating process can be performed into the data warehouse. However, if there if a large data volume to be updated from operational sources to data warehouse, this process takes time to be executed and all users should wait until the process ends to have access to both environments [11]. Therefore, the use of the near real-time approach and its variants are not suitable to comply with real-time requirements in a data warehousing environment. In other words, although the ETL process has suffered a lot of adaptations to be possible to apply it in a real-time environment, it is not yet sufficient to deal with real-time data warehousing environments.

3. RELATED WORK

In this section we show the proposed method, which deals with the extraction phase of ETL process in a real-time data warehousing environment. We consider the following three groups: 1) meeting real-time data warehousing requirements; 2) applied techniques to data extraction; 3) applied techniques to deal with data extracted from sensors. Regarding to meeting real-time data warehousing requirement, Bruckner et al. [26], Naeem et al. [27], Javed, M. and Nawaz, A. [28], YiChuan and Yao [29], MadeSukarsa et al. [30], Jain et al. [13], Jia et al. [31], Obali et al. [32], Mao et al. [33], Li and Mao [20] and Muddasir and Raghuveer [9] deals with low latency. Further, just Bruckner et al. [26] and Viana et al. [34] deal with availability requirements.

Although there are a lot of works that approach different aspects of real-time environment, few works deal with more than one of these requirements. Just Bruckner et al. [26] approaches the availability and low response time requirements, and Viana et al. [34] approaches the availability and scalability requirements. Just Guerreiro et al. [1] pointed out the fact that ETL process should deal with Big Data in a real-time data warehousing. Chieu, T. and Zeng, L. [35] deal with ETL process in a near real-time environment. However, authors do not show which real-time requirements are considered.

Regarding techniques to data extraction, in all works they did: 1) create a trigger inside the table of operational environment. By creating the trigger, it is possible to define the data of interest into trigger and store them into a target data warehouse whenever table receive an insert command; 2) fetching data of interest in log files of databases in an operational environment. It is possible to access these log files, fetching all data of interest and storing them into the target data warehouse. Regarding techniques to deal with data extraction from sensors, just Guerreiro et al. [1] show a means to extract data from it. However, the authors do not describe how the extraction is made. Moreover, to simulate the environment to generate and extract data, the authors made use of data stored into CSV files.

Although the aforementioned studies describe its methods to perform extraction phase of ETL process in real-time data warehousing environment, they differ from DEM. The main difference is that these methods do not decouple the operational environment and ETL process. In other words, for all methods, to apply ETL process, we should be intrusive, that is, we should access the operational source directly by using a wrapper, trigger, access a log file or other means to fetch data of interest. This fact causes a coupling between operational sources and ETL process, which in turn causes a performance decrease and a high cost to develop and maintain the data warehousing environment. Also, all works do not consider all requirements that DEM was built over.

4. DATA EXTRACTION MAGNET

In this section, we present DEM, a new and innovative method to comply with the extraction phase of ETL process in a real-time data warehousing environment. The Figure 1 shows the workflow of DEM and its components. This figure is composed of a representation of Operational Environment, Management Environment and ETL process. These components will be detailed in this section. To better explain the DEM workflow, we consider the following scenario: in the operational environment there are several heterogeneous data sources, which maintain operational data. It can be represented in Figure 1 by the Operational Environment component, which in turn is composed of operational sources. Moreover, data from operational sources are generated in real-time, for instance using sensors. These generated data should be extracted from the operational data sources in the ETL process.

Although there are a lot of methods to support the extraction phase of ETL process in real-time, all previous methods are based on traditional approaches. In other words, the data of interest are available to ETL process by using some CDC techniques (log files or triggers) and in intrusive way. By intrusive, these methods need to directly access the operational sources. This access is made through the wrapper connected to a log file or using triggers into relational tables. Also, these methods should deal with the details of connection to the data sources and the data heterogeneity of the operational sources. In an innovative way, DEM was developed to perform the extraction phase of ETL process in real-time in a data warehousing environment without the need to directly access the operational data sources, that is, in a non-intrusive way. To do this, DEM was built over the following properties:

- **Non-intrusive**: DEM does not access the operational sources to get data of interest using triggers, log file or some CDC techniques (that is why we call it non-intrusive). Instead, DEM is prepared to receive data of interest from operational sources by using the tag concept. So, this property causes a decoupling between the ETL process and the operational environment. Thus, this property aids to increase the performance of the extraction phase of ETL process, as this phase turns less expensive to be performed as DEM does not need to deal with the way to access the operational sources nor deal with the heterogeneity of data. Besides, DEM was designed to get data from operational sources using a specific format, which aid to solve the data heterogeneity problem. In other words, the operational source is responsible to share information about the structure and meaning of data using the specific format of DEM to receive data.

- **Tag**: DEM is a method to receive data (instead of seeking data into operational sources) from operational sources and make them available for the remaining of the ETL process. Specifically, DEM makes use of the tag concept. Tag is an indicator that an item of data (some data from operational sources like name, number of phones, city and so on) will be used in the future into a data warehouse to the decision-making process. Thus, the tag is placed on an item of data from operational sources. By using the tag, DEM is able to receive the item of data from operational sources and automatically make it available for some data warehouse that has interest in storing data that is associated with a tag.

- **Parallelism**: as an innovative way to think about the extraction phase of ETL process, DEM was built over a parallelism concept, that is, it makes use of a lot of parallel processes to support this concept. DEM is able to receive data of interest from the operational environment in parallel, which means that data of interest does not need to wait for a long time into a queue to be processed as the number of data that is received for ETL process increases.

By using these three aforementioned properties, DEM is able to extract data of interest from operational sources in a real-time non-intrusive parallel way. However, there should be a way to allow sharing data in a heterogeneous environment. Specifically in a data warehousing environment, the operational sources can be composed of a lot of data sources like relational databases, XML files, spreadsheets, NoSQL databases and so on. In the same way, a data warehousing environment is commonly composed of a relational database. To solve this problem, DEM uses a publish/subscribe system to deliver data from operational sources to the extraction phase of the ETL process [36], [37], [38]. This component can be seen in the Figure 1 by the Pub/Sub component.

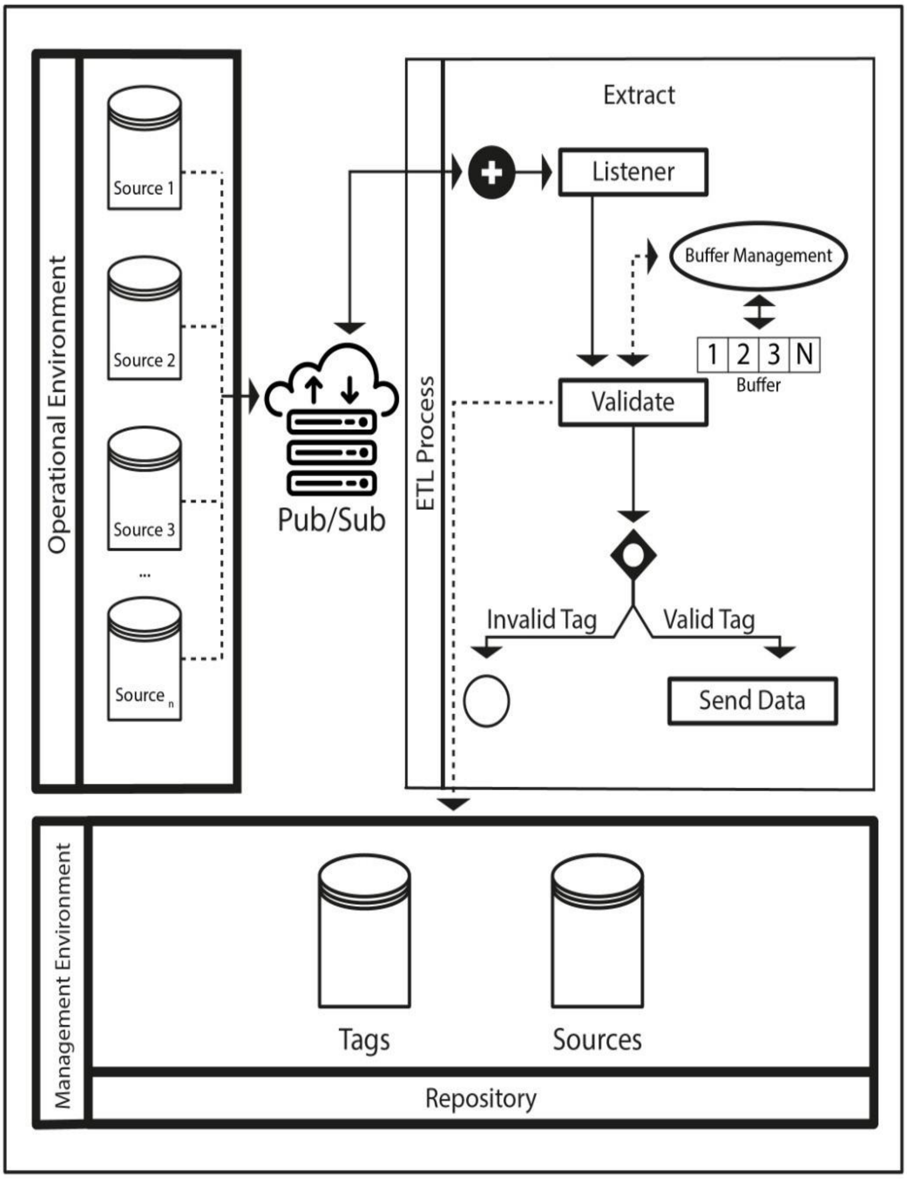

Fig. 1. Workflow of Data Extraction Magnet

The publish/subscribe system is composed by three main components:

- **Broker**: this component represents a cloud server, in which is responsible for receiving the data of interest from the publisher, storing them temporarily and making these data available to subscribers.
- **Publisher**: this component represents a data generator and in turn is responsible for sending data of interest to a broker. The data generator can be a data source, computer, smartphone, sensors, and any type of device that is capable to sending data to a broker.
- **Subscriber**: this component represents the entities that consume data sent by publishers. These entities can be relational databases, NoSQL databases, XML files and so on. The subscriber subscribes itself to the broker and expresses interest in receiving data from a

specific publisher. Whenever data is sent to broker by a publisher, subscriber is notified, and a new data is ready to be consumed. After that, the subscriber gets the data, sends a received notification, and waits for new data from a publisher.

By adopting the publish/subscribe system into DEM as a means of sharing data between the operational environment and ETL process, it is possible to integrate two different environments and overcome the problem of communication between heterogeneous environments [37]. Also, by applying the publish/subscribe system and those three aforementioned properties, we solved the problem to extracting data from one heterogeneous environment to another in real-time. Moreover, we consider that data generated by the operational environment has a strong link with the data warehouse that will store them in the future, and that is why we make the strong relation by using tag concept.

A. Repository of configurations

As previously pointed out, we need to configure tags so that we can receive data from the operational environment. So, these settings should be stored into a repository permanently and subsequently DEM can access them whenever needed to get all tags and in turn make data available for the remaining of ETL process. The repository of these settings can be seen in Figure 1 by the Repository component. The required configurations are: 1) tags; 2) data of interest from operational source and its respective tag. There are two repositories:

- **Tags**: this database stores the name of a tag that is defined by the administrator user.
- **Sources**: this database keeps relation between a tag (stored into Tags database) and a data of interest from operational sources.

It is worth highlighting that these configurations should be performed just one time by an administrator user. Once configured, DEM is able to access all repository whenever needed. This feature causes DEM act in a non-intrusive way and in turn contribute to reach the main feature of DEM.

B. Workflow of DEM

From the configurations defined by the administrator user, the following topics represents all components that DEM is built and in the order that these components are performed:

- **Operational Environment**: represents the operational data sources. In other words, this component represents a group of processes responsible for generating data in an operational environment. Also, it represents the environment that send (publishes) data to the broker to be consumed by the DEM. The operational source acts in an autonomous way and requests to send data to the data warehouse in a desirable moment. This fact allows the operational environment to adjust itself to the real-time requirements. So, the operational sources can opt to send data immediately to DEM and be sure that data will be prepared to be stored into the data warehouse immediately after data is generated in the operational environment. In this way, DEM considers that operational sources are important components to guarantee the real-time requirements. In other words, the operational source itself defines how will be defined the real-time requirements, due to the non-intrusive property.

- **Pub/Sub**: this component represents the publish/subscribe system, which is responsible to receive data from operational sources and make it available to be consumed by ETL process. It is important to note that this component represents just a way to share data between two environments. So, other techniques can be used to replace the publish/subscribe system. However, in our perspective, the publish/subscribe system is actually the better choice.

- **ETL Process**: represents the ETL process that performs extraction, transformation and loading phases. An important aspect to highlight is that the ETL process was built over a parallelism concept. In other words, whenever new data is ready to be received from the publish/subscribe system to ETL process, the Listener component takes the data and makes it available uniquely for the rest of ETL process. It causes the ETL process to work with a lot of data received parallelly.

    - **Extraction**: it is responsible to receive data from the operational environment. To do so, it is composed by a component named Listener.
    - **Listener**: this component connects to a broker and waits for new data notifications sent by the operational environment. When data notification is received, Listener gets this data and makes it available for the Validate process of extraction phase.

      It is worth to say that DEM should receive data in a specific standard format, that is, it requires that operational data sources send data to be stored into a data warehouse in a format that is recognized and accepted by the DEM. The standard data format is: ID(Data, Date, Time). From this component, all ETL process are started parallelly by using a lot of parallel processes. By doing it, DEM can deal with a lot of data of interest in parallel and this data does not need to wait into a queue to be managed.

    - **Validate**: it is responsible for validating the data that was sent by the operational environment. These validation processes include checking whether the data has a valid tag, that is, if there is a tag associated to the item of data sent from the operational environment. Other validation is the format of the sent data. If the validation is not true, the process is stopped. Otherwise, the data is sent to the Send Data component. This validation is performed automatically by DEM and by means of a seek into repositories.

    - **Buffer Management**: it represents a buffer that store temporarily the validations that had been made by the Validate process. Once validated, the validated data is stored into Buffer so that can be accessed in the future. If the validated data is already into Buffer, we do not need to access the repository and in turn provides a performance gain.

    - **Send Data**: this component represents the end of the extraction phase. Once data is prepared to be available for the remainder of the ETL process, this component is ready to send data to the target data warehouses that have interest in storing the validated data of interest.

5. EXPERIMENTAL EVALUATION

In this section, we describe the experimental tests used to validate the feasibility and the functionalities of DEM. This experimental test was performed in a dairy farming domain. We choose this domain because its characteristics are suitable and expected to be applied to DEM. In this domain, milk data from cows is collected three times a day, which are generated by sensors connected to the cows. These milk data values are extracted and used later for dairy farmers to making-decision process.

A. Performance Analysis

The elapsed time for sending data from operational sources to the ETL process is a good way to measure the performance and scalability of DEM. In this sense, we report the following measures for the elapsed time: minimum, maximum, average, median, percentile (p) and standard deviation. It is important to highlight that to be able analyze the median and percentile, we should keep the list of all performed operations sorted by response time from the fastest to the slowest.

The median is the value of time of the operation that is in the middle of the list of operations (p50). It means that median indicates the value of time spent to perform an operation that is in the middle of the list. In turn, the percentile is important to show the values of time spent to perform other operations that are above average time and p50. As the list of operations is sorted by average response time and in ascending order of average, by analyzing percentile, it is possible to expand the performance analysis to show how far the application can perform any operations in a suitable response time. So, the percentile is obtained from the value of spent time that corresponds to a 95% e 99% operation of the list of all performed operations.

The standard deviation (SD) is a measure that indicate the degree of dispersion of a data set. In other words, the standard deviation indicate how homogeneous are the collected data. We can consider how near zero is the standard deviation, the more homogeneous the data are. So, we can use the same unit of measurement to indicate the standard deviation, and, to this experiment, we will use seconds.

Beyond average, median, percentile and standard deviation, it is important to show the minimum and maximum elapsed time. The minimum elapsed time is important to show the best case in performing some operation. Otherwise, the maximum elapsed time is important to show the worst case in performing some operation.

We also gathered storage costs for maintaining Tag and Source databases. In fact, the Data Magnet only spent 0.2 MB.

Above all techniques to measure the behavior of DEM, we should test and compare its performance results against another commonly used technique to solve the extraction phase of the ETL process in a real-time data warehousing environment. In this way, we could show how good DEM is in comparison to the other techniques. In this sense, we will compare the results of performance and scalability of DEM against the trigger technique, which is one of the most used techniques to support the extraction phase of the ETL process.

We developed a prototype to allow us to generate milk data with the same characteristics and properties of real data. However, we use trigger to perform ETL process and in turn to get the measures.

B. Experimental Test

The experimental test was made using synthetic data, that is, the milk data produced by the experiment has the same semantic, properties and characteristics of data generated by real data. However, we can control the frequency and volume in which data is generated. The synthetic data was generated by using a software named Synthetic Data Generator (SDG), which was developed just to generate synthetic data. The experimental test was built to validate the availability, low response time and scalability of DEM when increasing the volume of data generated by sensors. By doing it, we can measure the elapsed time and in turn compare DEM against the trigger technique.

The data generated synthetically by SDG has the same schema of data generated by real domain, that is, data are generated into a JSON file, encompassing fields that represents the number of cow earring, date, time, and milk volume produced by the herd a real-time data warehousing environment. So, we can define values to these fields that represents the same range of values collected in the real case. By doing this, we can keep the same characteristics of generated data and their values and control the volume and frequency that milk data are produced. Thus, we can analyze the behavior of DEM and in turn we show availability, low response time, scalability requirements and its feasibility to apply it in environments with these characteristics. So, the section V-B1 depicts the settings to be able to perform the tests, section V-B2 shows the performed tests and results.

1) Workbench

To perform the experiments with synthetic data, we used the following set of software: 1) DEM, which was running in MacBook Pro with MacOS Catalina 10.15.7, 1.4 Intel Core i5, 8GB RAM; 2) SDG, which was running in Dell Inspiron 15R, Intel Core i5, 8GB RAM and Windows 10 64 bits SO; 3) Local MySQL database as a metadata repository, which was running in the same computer that was running DEM; 4) Eclipse Mosquitto as a publish/subscribe system, which is a free service provided by Eclipse. Moreover, the Internet used in this experimental test was a shared home Internet of 10MB and one router TP-Link TL-WR840N.

Regarding configuration of software, two main configurations were performed to allow us to run the tests. The first one is regarding DEM itself. To perform the experimental test with synthetic data, the settings are: 1) define all data of interest from operational sources, in this case, data generated synthetically by SDG; 2) define tags to be assign to data of interest; By doing it, DEM is ready to perform its task in a real-time parallel and non-intrusive way.

To generate synthetic data, it is needed to set how many synthetic cows, how many synthetic sensors and the frequency in which data will be generated by SDG. Thus, the generating of data simultaneously was guided by the proportion of 8 cows managed by 1 sensor, 2 synthetic sensors managing milk data of 16 synthetic cows, 3 synthetic sensors managing milk data of 24 synthetic cows and so on. So, the second configuration is the setting of SDG, which in turn allows define how many synthetic sensors, how many synthetic cows and frequency in which milk data are generated by each sensor.

2) Scalability of the Number of Sensors

The following three tests were performed into SDG, in which each test had its synthetic sensors sending data simultaneously: 1) 32 synthetic sensors managing milk data of 250 cows; 2) 63 synthetic sensors managing milk data of 500 cows; 3) 125 synthetic sensors managing 1,000 cows. The frequency of collecting of data was performed every thirty seconds. Further, all tests were performed using DEM itself and the trigger technique.

All performed tests reflect the current scenario of all dairy farming domains. In other words, currently, there are dairy farming with around five hundred cows in lactation. However, besides considering the current scenario, we went further, and we performed tests with 125 sensors to 1,000 cows in the worst case. By doing it, we can show the scalability of DEM as the number of sensors increases.

The Figure 2 shows the results of aforementioned measures obtained after performing the test 1. Also, this figure is composed by DEM and trigger measures. As we can see, the response time of DEM for all measures was lower than trigger. It means that throughout the test DEM behavior kept stumbling and its value of response time kept constant. We can prove it by analyzing Avg, p50, p95, p99 and SD measures.

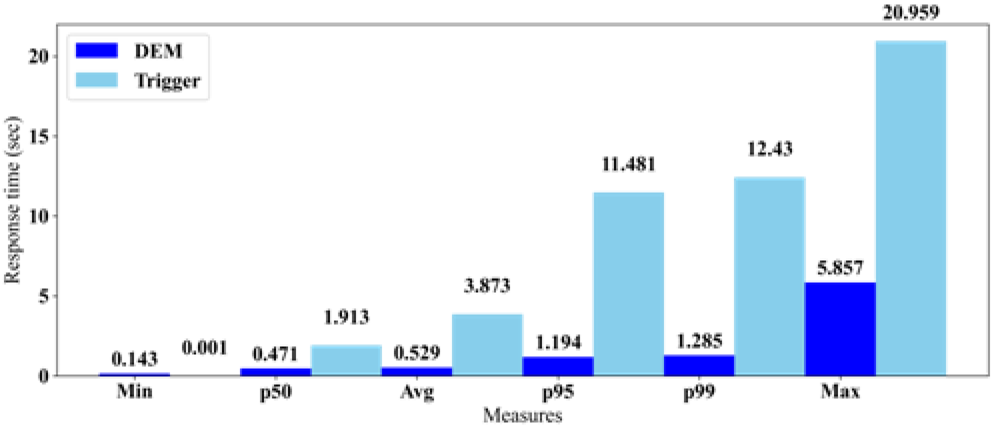

Fig. 2. Graphic of measures to show the response time of DEM and trigger for test 1. SD was 0.33 for DEM and 4.41 for Trigger.

Regarding to Avg value, DEM obtained 0.53 seconds whereas this value for trigger was 3.87 seconds. Regarding to p50, p95 and p99, the behavior of DEM is far better than trigger. The value of p50 for DEM was 0.47 seconds and for trigger was 1.91 seconds. The p95 and p99 value for DEM was 1.19 seconds and 1.29 seconds respectively, whereas the value of these measures for trigger was 11.48 seconds and 12.43 seconds. Regarding to SD, its value for DEM was 0.33 seconds whereas this value for trigger was 4.41 seconds. It means that SD value for DEM remained near zero. These values indicate that the elapsed time for DEM remained homogeneous from all collected data and for all performed tests.

Based on these analyzes, we can show a comparison of both techniques regarding performance gain and performance loss for measures. For almost all measures, DEM obtained a performance gain over trigger. For p50 measure, DEM obtained its lower performance gain, which was 306%, whereas for p99 measure, DEM obtained its higher performance gain, which was 867%. However, just for Min value DEM obtained a performance loss over trigger, which was 43%. The performance loss was caused by the overhead of managing parallel processes and tags.

The Figure 3 shows the results for the test 2. Also, this figure is composed by DEM and trigger measures. As we can note, the elapsed time of DEM for all measures was lower than trigger. It means that throughout the test DEM behavior kept stumbling and its value of elapsed time kept constant. We can analyze it by Avg, p50, p95, p99 and SD measures. Regarding to Avg value, DEM obtained 0.86 seconds whereas this value for trigger was 5.73 seconds. Regarding to p50, p95 and p99, the behavior of DEM was far better than trigger. The value of p50 for DEM was 0.49, whereas for trigger is 3.81 seconds. The p95 value for DEM was 1.21 seconds and for trigger is 17.23 seconds. The p99 value for DEM was 13.03 seconds for trigger was 20.07 seconds. Regarding to SD value for DEM, its value was 2.27 seconds, whereas this value for trigger was 5.57 seconds. It means that SD value for DEM remained near zero. These values indicate that the elapsed time for DEM remained homogeneous from all collected data and for all performed tests.

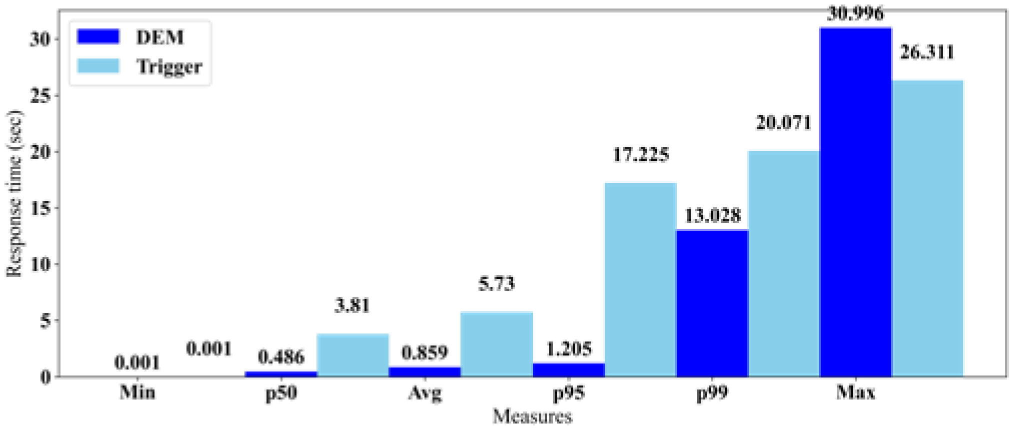

Fig. 3. Graphic of measures to show the response time of DEM and trigger for test 2. SD was 2.27 for DEM and 5.58 for Trigger.

Despite the Max value of DEM being higher than for trigger, it is not a strong measure to show how good one technique is in comparison to another technique. This is because it represents just a small fraction in relation to all collected data. Overall, by analyzing SD and Avg measures we can see that elapsed time for DEM kept constant and stable, whereas the elapsed time for trigger had a linear increase. Based on these analyzes, we can show a comparison of both techniques regarding the performance gain and the performance loss for each measure. For almost all measures, DEM obtained a performance gain over trigger. For p99 measure, DEM obtained its lower performance gain, which was 54%, whereas for p95 measure, DEM obtained its higher performance gain, which was 1,329%. However, just for Max value DEM obtained a performance loss over trigger, which was 17%.

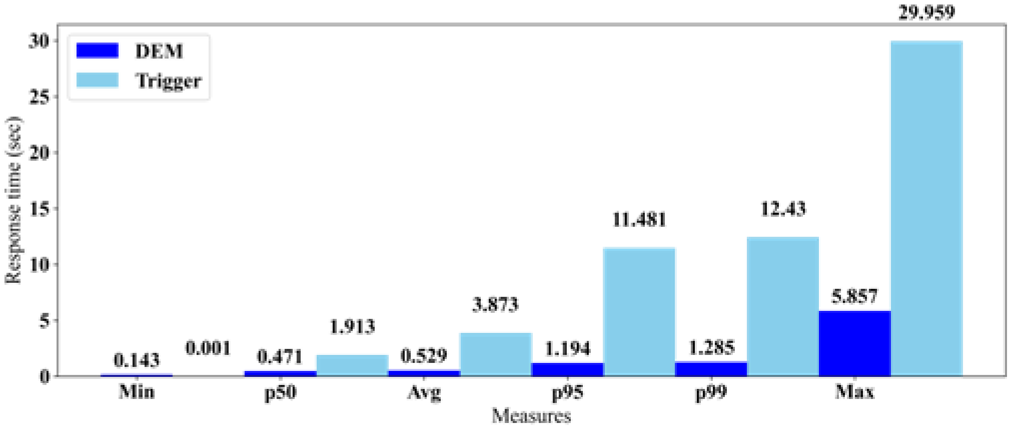

Fig. 4. Graphic of measures to show the response time of DEM and trigger for test 3. SD was 0.33 for DEM and 4.41 for Trigger.

The Figure 4 shows the results for the test 3. Also, this figure is composed by DEM and trigger measures. As we can note, the elapsed time of DEM for almost all measures was lower than trigger. It means that throughout the test DEM behavior kept stumbling and its value of response time kept constant. We can analyze it by Avg, p50, p95, p99 and SD measures. Regarding to Avg value, DEM obtained 0.47 seconds, whereas trigger obtained 3.87 seconds. Regarding to p50, p95 and p99, the behavior of DEM is far better than trigger. The value of p50 for DEM was 0.47, whereas for trigger was 1.91 seconds. The p95 value for DEM was 1.19 seconds, whereas for trigger, this value was 11.48 seconds. The p99 value for DEM was 1.29 seconds and for trigger was 12.43 seconds. Regarding to SD value for DEM, its value was 0.33 seconds, whereas this value for trigger was 4.41 seconds. As the SD value for DEM remained near zero, we can conclude that its response time remained homogeneous from all collected data and for all performed tests.

Despite the Min value of DEM being higher than trigger, in the same way of test 2, it is not a strong measure to show how good one technique is to the other. This is because it represents just a small fraction in relation to all collected data. Further, the Min value of DEM is 0.14 seconds, whereas this value for trigger is near 0.001 seconds, but even so, Min value of DEM is great. As we can note in test 2, by analyzing SD and Avg measure to test 3, we can see that the behavior and elapsed time for DEM kept constant and stable, whereas the elapsed time for trigger had a linear increase. So, regarding the performance gain and performance loss for almost all measures, DEM obtained a performance gain over trigger. For Max measure, DEM obtained its lower performance gain, which was 75%, whereas for p95 measure, DEM obtained its higher performance gain, which was 190,404%. However, just for Min value DEM obtained a performance loss over trigger, which was 117%.

6. Conclusion and Future Work

This paper presented the proposal of a new and innovative method to perform the extraction phase of the ETL process in a real-time data warehousing environment. Unlike the related work, DEM introduces the concepts of tags, parallelism, and non-intrusive properties into a real-time data warehousing context,and it shows a new way to accomplish the real-time ETL process. DEM defines the concept of real-time and discusses its applicability and feasibility for most applications that make use of the real-time data warehousing concept.

By applying non-intrusive property, DEM does not need to access the operational sources to get data of interest, conversely as is performed using trigger, log files and CDC techniques. Instead, DEM is designed to receive data of interest from operational sources. To do this, DEM makes use of another property called Tag. Basically, a tag is an indicator that an item of data from operational sources will be used in the future into a data warehouse to the decision-making process. So, this property causes a decoupling between the ETL process and the operational environment. Thus, this property aids to increase the performance of the ETL process, as the extraction phase turns less expensive to be performed, because DEM does not need to deal with the way to access the operational sources and also does not need to deal with the heterogeneity of the data.

From the experimental tests performed by using synthetic data, we could validate the low response time, scalability, and the decoupling between ETL process and operational environment. DEM provided low elapsed times even when the volume of data increased. Furthermore, DEM was better than the traditional trigger technique, where for almost all measures used to compare the two techniques, DEM showed lower response time and higher performance as data volume increased.

The main limitation observed during testing and evaluating the proposed DEM was the degradation of the performance for the max value. DEM showed in some tests a higher elapsed time for the max value then the trigger technique. This performance degradation for the Max value is because of the implicit overhead imposed by the whole environment, that is, two heterogeneous environments being integrated by a publish/subscribe system over the Internet. However, this performance loss represents a tiny quantity of all data gathered throughout the experiment. Also, for the most configurations and measures, the Data Magnet obtained a great performance gain over the trigger technique.

For future work, we intend to extend the performed tests by applying DEM into a real case of the dairy farming domain. Besides, for the load phase of the ETL process, we intend to build an innovative way to store data into a data warehouse, that is, once data is validated and it is prepared to be sent to target data warehouse, we will consider the tag concept and create a way to data warehouse recognize the data that is associated to a tag. So, this data can be sent directly to a target data warehouse.

REFERENCES

[1] G. Guerreiro, P. Figueiras, R. Silva, R. Costa, and R. Jardim-Goncalves, “An architecture for big data processing on intelligent transportation systems. An application scenario on highway traffic flows,” *2016 IEEE 8th International Conference on Intelligent Systems, IS 2016 - Proceedings*, pp. 65–71, 2016.

[2] P. Figueiras, R. Costa, G. Guerreiro, H. Antunes, A. Rosa, and R. Jardim-Goncalves, “User interface support for a big ETL data processing pipeline an application scenario on highway toll charging models,” in *2017 International Conference on Engineering, Technology, and Innovation: Engineering, Technology and Innovation Management Beyond 2020: New Challenges, New Approaches, ICE/ITMC 2017 - Proceedings*, 2017.

[3] M. A. Zamora-Izquierdo, J. Santa, J. A. Martínez, V. Martínez, and A. F. Skarmeta, “Smart farming IoT platform based on edge and cloud computing,” *Biosystems Engineering*, vol. 177, pp. 4–17, 2019.

[4] S. Fuentes, C. Gonzalez Viejo, B. Cullen, E. Tongson, S. Chauhan, and F. Dunshea, “Artificial intelligence applied to a robotic dairy farm to model milk productivity and quality based on cow data and daily environmental parameters,” *Sensors*, vol. 20, 05 2020.

[5] R. Mukherjee and P. Kar, “A comparative review of data warehousing ETL tools with new trends and industry insight,” in *Proceedings - 7th IEEE International Advanced Computing Conference, IACC 2017*, 2017.

[6] B. Yadranjiaghdam, N. Pool, and N. Tabrizi, “A survey on real-time big data analytics: Applications and tools,” in *Proceedings - 2016 International Conference on Computational Science and Computational Intelligence, CSCI 2016*, 2017, pp. 404–409.

[7] H. Chandra, “Analysis of Change Data Capture Method in Heterogeneous Data Sources to Support RTDW,” in *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8510574/

[8] F. Sabry and E. Ali, “A Survey of Real-Time Data Warehouse and ETL,” *International Journal of Scientific & Engineering Research*, vol. 5, no. 7, 2014. [Online]. Available: http://www.ijser.org

[9] M. N and R. K, “Study of Methods to Achieve Near Real Time ETL,” in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 2017, pp. 436–441.

[10] A. Wibowo, “Problems and available solutions on the stage of Extract, Transform, and Loading in near real-time data warehousing (a literature study),” *2015 International Seminar on Intelligent Technology and Its Applications, ISITIA 2015 - Proceeding*, pp. 345–349, 2015.

[11] Muddasir, Mohammed and Raghuveer, K, “CDC and Union based near real time ETL,” in *2nd International Conference on Emerging Computation and Information Technologies (ICECIT)*, 2017, pp. 1–5.

[12] A. Sabtu, N. F. M. Azmi, N. N. A. Sjarif, S. A. Ismail, O. M. Yusop, H. Sarkan, and S. Chuprat, “The challenges of Extract, Transform and Loading (ETL) system implementation for near real-time environment,” *International Conference on Research and Innovation in Information Systems, ICRIIS*, pp. 3–7, 2017.

[13] T. Jain, R. S, and S. Saluja, “Refreshing Datawarehouse in Near Real-Time,” *International Journal of Computer Applications*, vol. 46, no. 18, pp. 975–8887, 2012.

[14] M. Mesiti, L. Ferrari, S. Valtolina, G. Licari, G. Galliani, M. Dao, and K. Zettsu, “StreamLoader: An event-driven ETL system for the on-line processing of heterogeneous sensor data,” *Advances in Database Technology - EDBT*, vol. 2016-March, pp. 628–631, 2016.

[15] C. Kulatunga, L. Shalloo, W. Donnelly, E. Robson, and S. Ivanov, "Opportunistic Wireless Networking for Smart Dairy Farming," *IT Professional*, vol. 19, no. 2, pp. 16–23, 2017.

[16] M. Ryu, J. Yun, T. Miao, I. Y. Ahn, S. C. Choi, and J. Kim, "Design and implementation of a connected farm for smart farming system," *2015 IEEE SENSORS - Proceedings*, pp. 1–4, 2015.

[17] P. Figueiras, R. Costa, G. Guerreiro, H. Antunes, A. Rosa, R. Jardim-gonc¸alves, and D. D. Eng, "User Interface Support for a Big ETL Data Processing Pipeline," pp. 1437–1444, 2017.

[18] K. V. Phanikanth and S. D. Sudarsan, "A big data perspective of current ETL techniques," *Proceedings - 2016 3rd International Conference on Advances in Computing, Communication and Engineering, ICACCE 2016*, pp. 330–334, 2017.

[19] A. Sabtu, N. F. M. Azmi, N. N. A. Sjarif, S. A. Ismail, O. M. Yusop, H. Sarkan, and S. Chuprat, "The challenges of extract, transform and load (ETL) for data integration in near real-time environment," *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 22, pp. 6314–6322, 2017.

[20] X. Li and Y. Mao, "Real-Time data ETL framework for big real-time data analysis," in *2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics*, Lijiang, China, 2015, pp. 1289–1294.

[21] C. D. d. A. Ciferri, "Distribuição dos dados em ambientes de data warehousing: o Sistema WebD 2W e algoritmos voltados˜ a` fragmentação horizontal dos dados," Ph.D. dissertation, Universidade Federal de Pernambuco, 2002.˜

[22] R. Kimball, J. Caserta, R. Kimball, and J. Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning Conforming, and Delivering Data*, 2004.

[23] K. Kakish and T. A. Kraft, "ETL Evolution for Real-Time Data Warehousing," *Conference on Information Systems Applied Research*, vol. 5, no. 2214, 2012. [Online]. Available: www.aitp-edsig.org

[24] J. Langseth, "Real-Time Data Warehousing: Challenges and Solutions," 2004.

[25] C. Thomsen, T. B. Pedersen, and W. Lehner, "RiTE: Providing On-Demand Data for Right-Time Data Warehousing," *ICDE 2008*, vol. 00, pp. 456–465, 2008.

[26] R. M. Bruckner, B. List, and J. Schiefer, "Striving towards Near Real-Time Data Integration for Data Warehouses," *LNCS*, vol. 2454, pp. 317–326, 2002. [Online]. Available: http://www.ifs.tuwien.ac.at/ bruckner/pubs/dawak2002 data integration.pdf

[27] M. A. Naeem, G. Dobbie, and G. Weber, "An event-based near real-time data integration architecture," in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, 2008, pp. 401–404.

[28] M. Y. Javed and A. Nawaz, "Data load distribution by semi real time data warehouse," *2nd International Conference on Computer and Network Technology, ICCNT 2010*, pp. 556–560, 2010.

[29] S. YiChuan and X. Yao, "Research of Real-time Data Warehouse Storage Strategy Based on Multi-level Caches," *Physics Procedia*, vol. 25, pp. 2315–2321, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.phpro.2012.03.390

[30] I. MadeSukarsa, N. Wayan Wisswani, I. K. Gd. Darma Putra, and L. Linawati, "Change Data Capture on OLTP Staging Area for Nearly Real Time Data Warehouse Base on Database Trigger," *International Journal of Computer Applications*, vol. 52, no. 11, pp. 32–37, 2012.

[31] R. Jia, S. Xu, and C. Peng, "Research on real time data warehouse architecture," *Communications in Computer and Information Science*, vol. 392 PART I, pp. 333–342, 2013.

[32] M. Obali, B. Dursun, Z. Erdem, and A. K. Gorur, "A real time data warehouse approach for data processing," in *2013 21st Signal Processing and Communications Applications Conference (SIU)*. IEEE, April 2013, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/6531245/

[33] Y. Mao, W. Min, J. Wang, B. Jia, and Q. Jie, "Dynamic mirror based real-time query contention solution for support big real-time data analysis," *Proceedings of 2nd International Conference on Information Technology and Electronic Commerce, ICITEC 2014*, pp. 229–233, 2014.

[34] N. Viana, R. Raminhos, and J. Moura-Pires, "A real time data extraction, transformation and loading solution for semi-structured text files," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3808 LNCS, pp. 383–394, 2005.

[35] T. C. Chieu and L. Zeng, "Real-time performance monitoring for an enterprise information management system," in *IEEE International Conference on e-Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08*, 2008.

[36] J. Walkenbach, *Kafka the Definitive Guide*, 2010.

[37] M. Toshev, *Learning RabbitMQ*. Birmingham, UK: Packt Publishing, 2015.

[38] E. Onica, P. Felber, H. Mercier, and E. Riviere, "Confidentiality-preserving publish/subscribe: A survey,"` *ACM Computing Surveys*, vol. 49, no. 2, pp. 1–41, 2016.