Heliyon 9 (2023) e15728

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

# A non-intrusive and reactive architecture to support real-time ETL processes in data warehousing environments

Flávio de Assis Vilela [a], Valéria Cesário Times [b], Alberto Carlos de Campos Bernardi [c], Augusto de Paula Freitas [d], Ricardo Rodrigues Ciferri [d,*]

[a] *Federal Institute of Goiás, Brazil*
[b] *Federal University of Pernambuco, Brazil*
[c] *Embrapa Southeast Livestock, Brazil*
[d] *Federal University of São Carlos, Brazil*

## ARTICLE INFO

*Keywords:*
Data warehouse
Real-time
ETL
Data extraction
Data loading

## ABSTRACT

Nowadays, organizations are very interested to gather data for strategic decision-making. Data are disposable in operational sources, which are distributed, heterogeneous, and autonomous. These data are gathered through ETL processes, which occur traditionally in a pre-defined time, that is, once a day, once a week, once a month or in a specific period of time. On the other hand, there are special applications for which data needs to be obtained in a faster way and sometimes even immediately after the data are generated in the operation data sources, such as health systems and digital agriculture. Thus, the conventional ETL process and the disposable techniques are incapable of making the operational data delivered in real-time, providing low latency, high availability, and scalability. As our proposal, we present an innovative architecture, named Data Magnet, to cope with real-time ETL processes. The experimental tests performed in the digital agriculture domain using real and synthetic data showed that our proposal was able to deal in real-time with the ETL process. The Data Magnet provided great performance, showing an almost constant elapsed time for growing data volumes. Besides, Data Magnet provided significant performance gains over the traditional trigger technique.

## 1. Introduction

Nowadays, strategic decision-making has been crucial in several domains, such as smart farm systems, health care systems, and road control systems [1,2]. Data are provided by operational sources, and they are handled by OLTP (Online Transaction Processing) systems. Furthermore, these data are disposable in several shapes. Hence, there are different ways to maintain the data used by the strategic decision-making. Examples to maintain data of interest are relational and non-relational databases, text, XML or JSON files, spreadsheets or anything else formats that store data [3,4] [3,4].

Operational data can be produced in several ways, among which by: 1) by users interactions with online systems, for instance Web systems; 2) using batch processing systems; 3) using sensors, for example sensors connected to human body; 4) automatically, using satellites, which send high volume of data to stations in the earth (i.e., climate data). Indeed, the strategic decision-making is made

[*] Corresponding author.
*E-mail addresses:* flavio.vilela@ifg.edu.br (F. de Assis Vilela), vct@cin.ufpe.br (V.C. Times), alberto.bernardi@embrapa.br (A.C. de Campos Bernardi), augusto.freitas@estudante.ufscar.br (A. de Paula Freitas), rrc@ufscar.br (R.R. Ciferri).

https://doi.org/10.1016/j.heliyon.2023.e15728
Received 12 October 2021; Received in revised form 24 May 2022; Accepted 19 April 2023

Available online 26 April 2023

gathering data from the operational sources. Therefore, the gathered data must be treated by the ETL process to help the strategic decision-making. The ETL workflow deals with the gathered data to comply with extraction, processing, cleaning, integration and loading of the data. Therefore, the data for strategic decision-making are delivered to the data warehousing system and managed using an integrated, dimensional, and homogeneous database, that is called data warehouse [5,6,3].

A data warehousing system is a core part of a business intelligence systems and in turn it supports the decision-making process. A data warehousing environment has architectures, systems, algorithms, and tools that enable to gather data from operational sources to store the data into the data warehouse [7]. Additionally, the main characteristic of the data warehousing is regarding to the historically way of storing data. Due to the accumulated data during a prolong period of time, the volume of data that is managed is enormous and can increase quickly. In fact, the volume of data stored into data warehouse can easily reach petabytes of storage [7,6,8].

### 1.1. Traditional extract-transform-load process

The ETL is organized by a workflow with several tasks, which are partitioned in three phases: 1) the first phase is called Extraction. The main goal of this phase is to gather the data from operational sources. This is performed using wrappers, triggers, or log files. For this purpose, it is established a communication to operational data sources. Each data source has its own access path and data format; 2) other phase is named Transformation. This phase performs data cleaning, data integration, data filtering, data transformation, and data processing. In this second phase, there are removed the conflicts of values, semantic, and structural conflicts, such as inconsistent values, synonyms, and namesake. The transformed and cleaned data are stored in the data staging area; 3) the last phase is called Loading. This phase performs the load of the data of interest to the target data warehouse [9,10]. The ETL workflow aims to store operational data in a consistent and integrated way [9,11].

The traditional ETL sends data from operational sources to the data warehouse in a given period, that is, once a day, once a week, monthly, once a half-year, or yearly. Besides, the ETL workflow is performed in a loading time window in off peak hours, for example, during the night window [9,3,12]. During the loading time window, the informational environments (OLAP systems) should be offline to do the ETL process [13]. While making changes in the data, the on-line analytical processing systems can not get up-to-date data. Therefore, the ETL workflow must perform as soon as possible during the time window for loading data. As Muddasir and Mohammed [8], using this time window is only acceptable for organizations that enable this feature, for which data can be updated less often. So, the latter data does not take so much importance in determining historical patterns. In general, the use of the time window for data modification is adequate for most applications. In fact, the loading time window has been widely applied in real systems with less cost when using the data incremental updating approach [14].

### 1.2. New demands for real-time data warehousing environments

Real-time ETL processes must guarantee three essential requirements: low response time, high availability, and scalability. Besides, it is also desirable that the following requirements be guaranteed: recoverability and decoupling between operational data sources and the ETL process. Latency and response time (or deadline) are concepts that frequently are applied as the same, but they are different [15,16]. Latency is the time that a task is waiting in a queue to start to be processed. On the other hand, response time is the total elapsed time that a task takes to be performed, including latency time and implicit overhead time for data exchange.

The feature of data update of the ETL process can negatively impact the analysis of data for applications that require real-time processing [17]. Indeed, the results reported by the On-Line Analytical Processing queries, which were processed by the data warehouse during the day, in general can retrieve an wrong value in comparison for the data that are effectively stored into the data warehouse. This occurs because the target database is updated only once, twice, or few times during the day. On the other hand, the operational data sources produce data in high frequency, night and day [18,8]. Hence, this feature became difficult to apply the conventional ETL workflow in real-time DW environments. Furthermore, Mesiti et al. [19] argue that, nowadays, the high number of distinct types of sensors with even lower cost used in several different contexts were the motivation to offer many services that need data gathered from sensors to the decision-making process. These services can be natural disasters monitoring, road congestion, temperature, atmospheric pressure, cattle automatic weighing in a farm and so on. However, there is a need to offer new and efficient ways to perform the ETL process, so that it can be possible to handle the high frequency of the data flow produced by sensors. Thus, the current solutions for ETL process are not yet mature enough to appropriately deal with real-time requirements.

Several applications have been emerged to handle strategic decision-making data. In particular, these data are generated very quickly, which needs gathering data considering real-time requirements, like performed by environments that handle data from sensors [19] and health care systems [20]. Furthermore, specific applications, such as smart farm systems [21–23,2] and systems that handle road traffic data [24], not necessarily produce a huge data volume. However, these applications still need data be available immediately, according to real-time requirements, for the DW to properly accomplish to strategic decision-making [14,25].

Data are updated in the data warehouse using three main ETL approaches, which seeks to reduce the time to make the data delivered to the decision-making process: 1) the first approach is called on-demand. In this approach only the steps of loading and extracting data are carried out. The data is sent to the DW without performing data cleaning, data integration, data filtering, data transformation, and data processing. For each requested OLAP query, the remaining data operations (i.e., data cleaning, data filtering, data transformation, data integration, and data processing) are performed in an ad-hoc manner and results are issued for the user. Therefore, the core of the ETL is performed only at query processing time. Despite being fast, the on-demand approach has a drawback, since it requires a huge elapsed time to deal with On-Line Analytical Processing queries; 2) the second approach is called near real-time: here, data is delivered to the DW more quickly than the traditional ETL process. More specifically, many companies require that the data be produced in a lower time. Therefore, the ETL process is not indeed performed in real-time, but several times per day. In this scenario, it occurs a short latency, such as a couple of seconds. According to Sabry and Ali [6], Muddasir and Mohammed [8], Langseth, J. [26], Kakish and Kraft [11], there are several techniques commonly used to accomplish ETL workflows to meet the goals of the near real-time properties. The techniques are: *Direct trickle feed*, *External real-time data cache* and *Trickle and flip*. The main difference for the techniques is the way they manage historical and real-time data. However, all techniques are made just nearly real-time and using negatively an strategy that is intrusive and non reactive. By a strategy that is intrusive, we use the following definition: a strategy is intrusive when it needs to directly connect to the data sources to gather data and explicitly cope with the connection to the operational sources. By non reactive strategy, we use the following definition: a non reactive system does not perform any task (i.e., insert, update, delete or anything else) automatically; 3) the third approach is called real-time: this approach differs from near real-time approach by storing the operational data for the DW considering real-time requirements. Thus, for real-time approach, data are produced continuously by the operational data sources, which are sent and published in the DW as soon as possible an insert event happens [5,25,8,9,27]. In this way, the real-time system concepts have been incorporated in the design of DW with real-time properties to accomplish the immediate reaction to events occurring for the operational data sources. In a real-time system, not only the results generated by these systems are fundamental, but also the elapsed time is a major fundamental feature [15].

The basic attempt to reach the real-time requirements in ETL processes is to increase the frequency of updating for the DW. So, ETL workflow is performed several times a day using an idle time of the system. This strategy simulates the near real-time approach [18,5,9]. This solution can produce three main problems: 1) the operational data sources aid users that carry out operations frequently. By default, the conventional ETL are performed in an intrusive manner. It means that the convention ETL connects to the sources to gather data of interest. This feature produces a query overloading at the moment that conventional ETL process access the operational sources to gather data. Hence, this feature causes a performance degradation of the operational data sources; 2) DW is applied to users that carry out On-Line Analytical Processing queries, which are complex and can result in a large volume of data, resulting in an huge elapsed time to be performed. So, DW in general are not available to be changed until the query processing finishes; 3) when operational (OLTP systems) and information (OLAP systems) environments are idle, it is possible to update data inside DW. If a huge volume of data is ready to be updated, it can take a long time to be processed and the system becomes unavailable until the process ends [8]. Indeed, by using near real-time properties is not enough to deal with real-time requirements in DW systems.

It is important to highlight that some concepts are not homogeneous in related work concerning real time DW systems. The first one is the real-time concept. A real-time environment should respond events, respecting the temporal restrictions imposed by the applications. The correct running of these systems depends on the results of computations and of the time needed to be performed [28]. Moreover, there are two different terms that it is applied to classify the real-time concept [15]: 1) hard real-time: the failure to comply with the constraint of time can cause an actual harm of catastrophic consequence; 2) real-time that are soft: That is, the failure that comply with the limits of time does not produce strong loses. Moreover, the systems that are developed by using soft real-time usually do not need to prove the requirements of real-time performance. In fact, complying with all response time is not sufficient and it is not the only aspect to be considered. One example of systems that can apply soft real-time is on-line systems. These systems can perform a lot of activities in real-time. On-line system is a computer system whose operation is based on data that is gathered during its own operation (as opposed to batch systems). In other words, an on-line system is a computer system where the input data is gathered from a terminal.

However, an implicit overhead to exchange data occurs in on-line systems, such as between operational data sources and the data warehousing environment. It is caused mainly by to the cost to perform data cleaning, data filtering, data transformation, data integration, data processing, data extraction, and data loading, but also caused by the limitations of transfer rate, hardware limitations, the use of the internet, and the overhead of the network protocol. The costs enhance the time that data are delivered to the DW. Therefore, soft real-time in on-line systems could not guarantee an elapsed time lower than the implicit overhead time for data exchange, even being processed in real-time.

### 1.3. Contributions

This paper presents an innovative manner to accomplish real-time ETL workflows for DW systems, which is called Data Magnet. The core of the Data Magnet uses both the non-intrusive property and also the reactive property. Also, DM has the following properties: low response time, on-line and soft real-time data processing, scalability, high availability, and decoupling the ETL workflow and the sources. Furthermore, according to the common techniques to perform the ETL workflow, the DM provides a great improvement to accomplish the ETL workflows considering real-time requirements.

The DM architecture focus on the loading and extraction steps of the ETL workflow. The Data Magnet does not handle the transformation step of the ETL workflow as the workflow for the transformation step is highly dependent on the requirements of the target application and does not influence the loading and the extraction phases. So, the DM can receive the data from operational data sources according to requirements that are real-time and send data for target DW repositories as soon as possible. In a nutshell, The Data Magnet can perform all the extract, load, and transformation steps of ETL workflows.

Regarding real time requirements, we consider soft real-time to build the Data Magnet [15]. In this sense, Data Magnet tries to comply with the deadline for processing a given task, as the non-compliance with the deadline requirement is not so important to DW systems designed to meet real-time properties. Moreover, DM does not know the schema of the sources. Instead, the sources

**Table 1**
Comparison between functionalities of related work. A reactive system reacts to an insert event in the operational data sources and receives automatically the data. An intrusive system requires access to the operational data source by using a wrapper, trigger, log file, or other technique to gather data of interest.

| # | Author | Year | Real-time | Reactive | Intrusive | ETL |
|---|---|---|---|---|---|---|
| 1 | Bruckner et al. | 2002 | A | No | Yes | ETL |
| 2 | Viana et al. | 2005 | S and A | No | Yes | ET |
| 3 | Naeem et al. | 2008 | RT | No | Yes | EL |
| 4 | Chieu, T. and Zeng, L. | 2008 | RT | No | Yes | EL |
| 5 | Santos and Bernardino | 2008 | RT | No | Yes | L |
| 6 | Chen et al. | 2010 | RT | No | Yes | L |
| 7 | Javed, M. and Nawaz, A. | 2010 | RT | No | Yes | E |
| 8 | YiChuan and Yao | 2012 | RT | No | Yes | EL |
| 9 | MadeSukarsa et al. | 2012 | RT | No | Yes | EL |
| 10 | Jain et al. | 2012 | RT | No | Yes | EL |
| 11 | Jia et al. | 2013 | RT | No | Yes | EL |
| 12 | Obali et al. | 2013 | RT | No | Yes | EL |
| 13 | Valencio et al. | 2014 | S | No | Yes | E |
| 14 | Mao et al. | 2014 | S | No | Yes | E |
| 15 | Freudenreich et al. | 2014 | S | No | Yes | E |
| 16 | Li e Mao | 2015 | RT | No | Yes | L |
| 17 | Guerreiro et al. | 2016 | A | No | Yes | TL |
| 18 | Muddasir e Raghuveer | 2017 | RT | No | Yes | EL |
| 19 | Machado et al. | 2019 | RT, A and S | No | Yes | ETL |
| 20 | Muddasir and Raghuveer | 2020 | RT | No | Yes | L |
| 21 | Thulasiram and Ramaiah | 2020 | RT | No | Yes | L |
| 22 | Vilela et al. | 2021 | RT, A and S | Yes | No | EL |

act to ensure real-time requirements, that is, delivering data to the data warehouse whenever they want and in general as soon as possible.

This article is divided as follows: Section 2 shows related work. Section 3 details the architecture of the Data Magnet and its workflow. Section 4 presents the experiments performed to assess the Data Magnet. Section 5 highlights future work and finishes the article.

## 2. Related work

As for methodology, we were guided by the principles of DSRF (Design Science Research Framework) method [29]. In particular, the first step used by the DSRF method consists of identifying the research problem and to give a justification for the value of the proposed solution. For this purpose, we performed a wide revision to identify the best solutions of the issue being investigated in this paper, that is real-time ETL processes for data warehousing environments. After the revision of the literature, we were able to fulfill Table 1, which in turn shows the most important related work about the research problem and the proposed solution. This table is composed by the Author, Year of publication, the real-time requirement dealt by the related work (i.e. (A) Availability, (RT) Response Time, (S) Scalability), if the work is related to the Reactive property, if the work is related to the Intrusive property and which ETL phases the work investigated.

There are several studies that investigated the issue of ETL applied in real-time to DW systems. The related work was summarized in Vassiliadis and Simitsis [30], Penzlin et al. [31], Farooq and Sarwar [32], Tank et al. [33], Kakish and Kraft [11], Sabry and Ali [6], Revathy et al. [34], Ferreira et al. [35], Wibowo, A. [12], Gajanan Mane [13], Muddasir and Raghuveer [9], Sabtu et al. [14], Bouaziz et al. [36], Phanikanth and Sudarsan [25], and Chandra, H. [5]. However, we focus here on proposals of architectures to support real-time ETL processes. So, we classified related work in three groups: 1) studies that deal with real-time requirements in DW systems; 2) studies and techniques to data extraction.

Regarding real-time ETL processes requirements, Bruckner et al. [37], Naeem et al. [38], Javed, M. and Nawaz, A. [39], Yao and Yi Chuan [40], Jain et al. [20], MadeSukarsa et al. [41], Obali et al. [42], Jia et al. [43], Li and Mao [18], Mao et al. [44], and Raghuveer and Muddasir [9] focuses on slow elapsed time. Besides, only Viana [45] and Bruckner [37] focused on the availability requirement.

Although there are studies that investigated distinct aspects of real-time ETL processes, few studies treated jointly two or more of requirements required by real-time environments. Only Bruckner [37] investigated slow response time and availability, and Viana et al. [45] investigated scalability and availability requirements. Guerreiro et al. [46] focused on the scalability requirement and mentioned that ETL process should comply with requirements of Big Data in real time DW systems. Chieu, T. [47] investigated ETL in a near real time DW systems. In a negative way, Chieu, T. [47] do not present which real-time requirements were investigated. Chen et al. [48] focuses on continuous loading and consequently on availability.

Regarding techniques to data extraction, the studies mainly: 1) defined a trigger for a table of the operational data source. Once the trigger is created, we can define the data of interest to be gathered by the trigger and store the data into the target database using a command of insertion; 2) fetch data and store them inside a log file. By doing this, the log file can be accessed later to fetch data and then put the data inside the proper database. For example, Valêncio et al. [49] uses triggers for managing inserted data inside the data source to the proper database. Jain [20] and Machado et al. [50] used log files to extract data of interest that is inserted into operational data sources. So, these data are stored into a temporary database with the same scheme of the DW.

Naeem [38], Chieu, T. [47], YiChuan [40], Obali [42], Freudenreich [51], Mao [44], Mao and Li [18], Guerreiro [46], Raghuveer and Muddasir [52] and Ramaiah and Thulasiram [53] focus on loading phases of ETL process. The authors use real-time partition techniques to store separately historical and real-time data. These techniques allow real time datasets to be stored and managed in a distinct way of how historical data is manipulated. By applying real-time partition techniques, a bigger and more complex structure is needed to be able to maintain two distinct databases. Further, real-time partition techniques demand new tasks to be applied to synchronize both databases.

Despite all studies described previously that investigated real-time ETL processes used for gather data to data warehouses, they have significant differences from the Data Magnet. Previous studies did not comply with the non-intrusive property and reactive property. Therefore, previous studies are intrusive, so, they connect the sources to get data by using wrappers, triggers, log files, or other techniques. Then, there are a hard coupling among operational sources and ETL workflows. This characteristic degrades the elapsed time and provides a enormous overhead to deal with the DW system. Furthermore, each previous study does not deal with jointly the whole requirements for real-time applied to ETL workflows, that is: high scalability, low response time, and also very high availability.

Another negative and surprising aspect of the related work is that they do not cite each other. After exhaustive systematic revision, we notice that the most recent studies that investigated real-time ETL processes for use specifically in data warehousing environments do not know about previous studies. This negative aspect makes it difficult to identify the evolution of the solutions for real time ETL workflows, and also the evolution of techniques, architectures, methods, methodologies, and general studies.

Despite the drawbacks highlighted for the related work, we can note that the architectures developed by Valêncio et al. [49], Muddasir and Raghuveer [9] and Machado et al. [50] can be considered the best solution for extracting data for ETL workflows. Valêncio et al. [49] made a dynamic process to create a trigger in operational data sources to gather all changed data. Muddasir and Raghuveer [9] use the Direct Trickle Feed technique to increase the frequency in which the ETL process is performed. They made use of critical and non-critical data analyzes into a log file to decrease the number of rows and indeed provide a performance improvement. Lastly, Machado et al. [50] made use of a log file to access the data of interest, and stored them into a table in memory which in turn can improve the performance. Also, the authors demonstrated how can be applied the architecture by using Kafka, Beam, and Spark Streaming.

Regarding the loading phase, Li and Mao [18] can be considered as the best solution for ETL workflows. The proposal uses a dynamically mirror technology to prevent the overhead between OLTP updates and OLAP queries. Indeed, their proposal is based on a External Real-Time Data Cache technique to be able to deal with historical and real time data at same time. Real time datasets are stored into repositories that have the same schema of the repository that stores the historical data. The synchronization of both repositories is made in batch mode after a certain system condition. Also, the architecture can improve the freshness of query results by obtaining query results from both repositories.

From the aforementioned comparative studies and the state-of-the-art described, all these things contributed to establish the following Project Requirement (PR):

- PR-1: Real-time ETL processes must guarantee low response time, high availability, scalability and recoverability.
- PR-2: Real-time ETL processes should not use the traditional techniques for the extraction phase, such as wrappers, triggers, log file and delta file.
- PR-3: Real-time ETL processes should allow data sources to act in an autonomous way and request the sending of data of interest to the target data warehouses whenever it wants. Therefore, data sources should define their own real-time constraints.

The second phase used by the DSRF method is performed to indicate the goals of a solution for the detected problems in the previous step. For this purpose, we defined the following Project Principals (PP) to at least reduce the challenges that were found from the systematic revision of the literature. The principals are:

- PP-1: Real-time ETL processes should allow on-line processing.
- PP-2: Real-time ETL processes should be non-intrusive, allowing decoupling between the operational sources and the ETL process.
- PP-3:Real-time ETL processes should share data between the operational sources and the ETL process by means of a publish-subscribe system.

The third phase of the DSRF method is to make a project and propose a solution. From the project requirements and the project principles, We will detail forward the DM architecture to solve the research problem. Lastly, the fourth step of the DSRF method is to demonstrate how the architecture was implemented and tested. So, in the next sections we will argue how the project requirements were met by the Data Magnet and we will present the experimental workbench to certificate the Data Magnet architecture.

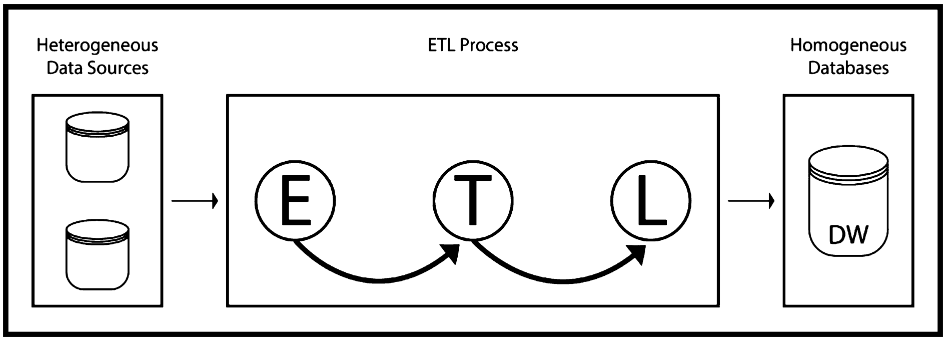


**Fig. 1.** Workflow of the ETL process.

## 3. The Data Magnet

We present in this section the architecture of the DM, an optimized manner to deal with real-time ETL processes requirements for data warehouses. Fig. 1 illustrates the workflow of the ETL process. There are several operational data sources that store data of interest for decision-making processes. The operational data should be gathered from the operational sources, then transformed (i.e., perform data cleaning, data filtering, data transformation, data integration, and data processing), and finally delivered to DW in consistent and integrated manners. After that, the data will be available to be used. For this purpose, the Data Magnet gets a crucial function to deliver data to DW systems in real-time.

Although there are many techniques and studies that proposed solutions for real-time ETL processes, all the current proposals are based on traditional solutions. Then, the data are gathered by applying CDC (Change Data Capture) techniques, such as, log files, delta files or triggers. Besides, the current solutions are non-reactive and intrusive. By intrusive, it means that the techniques access the data sources. This access is performed, for example, using wrappers and log files or using triggers defined for the tables of the sources. Also, the current techniques must comply with the diversity of the sources and the connection to the data sources. On the other hand, the non-reactive property enforces that all techniques do not need to react immediately to deliver data to the DW when a given operation of insertion happens inside data sources.

On the other hand, the Data Magnet was designed to comply with real-time ETL processes without requires the connection to operational sources. In other words, data extraction from sources is made using the non-intrusive property, reacting to an operation of insertion that happens in the sources. Also, the Data Magnet receives data, and sends the data to the transformation phase and then stores the data into a given data warehouse.

For guaranteeing the appropriated treatment for real-time ETL processes, the Data Magnet was designed based on the following properties:

- **Non intrusive**: The DM does not connect to the data sources aiming at gathering data. Therefore, there is no need to use log files, triggers, or CDC techniques (i.e., non-intrusive techniques) in the operational data sources. Instead, the DM is designed to gather data throughout tags. It enables a decoupling among the operational data sources and the ETL process. Therefore, the elapsed time produced by the ETL workflow is increased, since the cost of the extraction becomes lower.
- **Reactive**: The DM reacts to insertions in the sources and gather datasets. For this purpose, the DM uses a publish subscribe software to support sending of the data to the ETL process. By applying it, the data sources work as a publisher. In this sense, they connect to a broker (i.e., server) and send (i.e., publish) data. Meanwhile, the DM access the broker and then acts like a subscriber. In other words, the DM connects to the broker and gets data that was sent from the publisher. It is important to note that the insert operation from operational data sources can occur whatever the moment by a trigger or any other way to insert data. Also, the system in the operational side needs to send data for the broker to allow DM to receive the data and perform its tasks. Data Magnet does not produce any data and just receives the data produced by operational data sources.
- **Tag**: DM is designed to get data using data sources, instead of search data into the data sources, and storing data in the target DW. For this purpose, the DM is based on the concept of tags, which indicates that a data item of the interest (i.e., relevant data of data sources, such as name, age, state, address columns in a relational table) will be stored into the data warehouse. Tags are used to label each data item that is stored in operational sources. By using tags, the Data Magnet can gather the data and identify target DW that need to manage the data.

By using the aforementioned properties, the DM can carry out real-time for ETL in a reactive and non-intrusive fashion. However, it is necessary to allow sharing data among heterogeneous environments. In general, the operational data sources are composed of different kinds of data sources, such as NoSQL databases, relational databases, spreadsheets, and XML files. To allow sharing data among heterogeneous environments, the Data Magnet is based on a publish/subscribe system and requires a specific format to send data to ETL workflows. As a consequence, DM can integrate 2 distinct systems and give a solution to the problem of heterogeneity [54]. Another benefit of the adoption of the publish-subscribe software is that the DM can perform a lot of tasks in parallel. It means

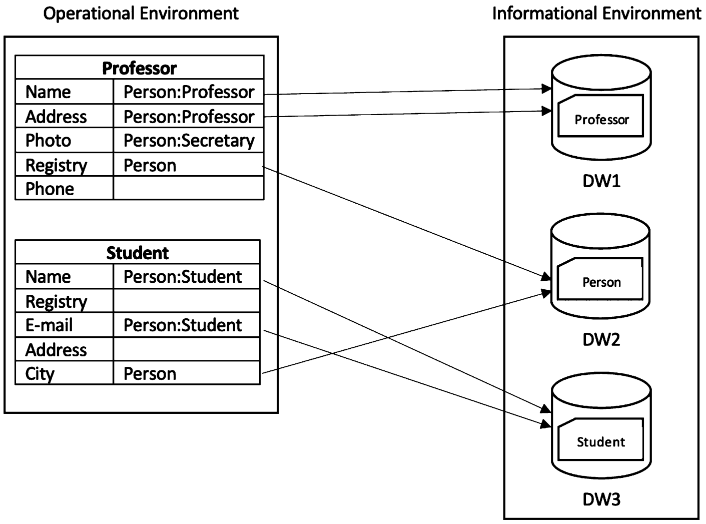


**Fig. 2.** Basic example of the Data Magnet.

that, once a data does not need to be maintained into a queue, nor into a temporary table or using a pooling approach to allow the subsequent processing of the data, we can obtain a performance gain for the whole process.

The publish subscribe software is composed by the following components:

- **Publisher**: the component is formed by a set of sources and its main goal is send data to the broker. These sources can be of several kinds, such as a sensor, a computer, or other kind of device that can send data to the broker.
- **Broker**: a service in the cloud that deals with the gathering of the data in the publisher, and after storing data locally and get reading data to be read later by subscribers.
- **Subscriber**: this component is composed by the target sources that consume data sent by publishers. This component subscribes to the broker and manifest that will gather data using a given publisher. If the publisher sends data to the broker, the subscriber is notified by the broker, and the novelty data can be gathered by a given subscriber. Then, a subscriber consumes the information, delivers a notification of receipt, and awaits that more recent data arrives from a publisher. In DM architecture, the subscriber represents the extraction of data of interest in the ETL workflow.

Before we detail DM architecture, the following scenario should be considered to better understand the Data Magnet: in a hypothetical academic data warehousing environment, the sources maintain the systems that handle data from professors and also students. The data of professors are stored into a database named University, inside a table named Professor and it has five attributes, i.e., name, address, photo, registry, and phone. Also, this database has a table named Student and it has five attributes, i.e., name, registry, e-mail, address, and city. In the data warehousing environment, there are three data warehouses. The first one (DW1) only stores data from Professor, the second one (DW2) stores data from Person (Professor and Student) and the third one (DW3) only stores data from Students. The data should be gathered using data sources and stored into a DW according to their interest in the strategic decision-making. This is performed by means of the Data Magnet.

However, to make use of the Data Magnet, some settings should be performed by the administrator user, so that all tasks of the Data Magnet can be accomplished in a non-intrusive and reactive way. To illustrate this scenario, we will consider the Fig. 2 to better explain the concept of tags introduced into the Data Magnet. The first setting defines the data of the sources that will be stored into the data warehouses. For this purpose, the administrator user should create tags and associate them to data from the operational data sources. In fact, there is no need to label each data (for example, each column of a relational table). Only data that is of interest of the strategic decision-making must be labeled, more specifically, the data of interest in the operational data sources. A tag can be created as a single or hierarchical tag. Single tags indicate that a data does not be associated to a group of tags and in turn it does not be related to subsets of the data. Hierarchical tags are used to define subsets of the data for a given data. For instance, the tags Professor and Student can be used to define subsets of data for the tag Person. Therefore, a Professor is a Person. In the same way, a Student is a Person. However, Person does not necessarily store only Professors and Students, since it can additionally store other kinds of Person, such as Technical employee or a Secretary.

In the following scenario, the administrator user can firstly create the simple tag Person and then create the hierarchical tags named Person:Professor and Person:Student. Once created, these tags should be associated with the operational data of interest. In this sense, the tag Person:Professor should be associated with the Name and Address columns of the Professor table. Also, the tag Person:Student should be associated with the Name and Email columns of the Student table, whereas the tag Professor should be associated with the Registry column from the Professor table and City column from the Student entity. Note that there are columns in the Professor and Student tables that are not data of interest and, therefore, these columns are not labeled using tags (i.e., Phone column in Professor table and Registry and Address columns in Student table).

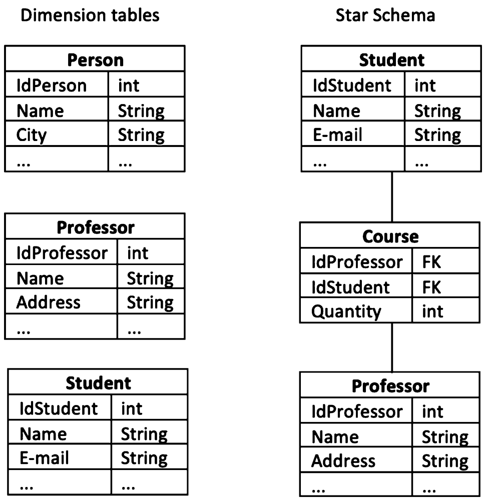


**Fig. 3.** Basic example of the star schema of the hypothetical scenario.

Moreover, the administrator user should define which data warehouses (DW1, DW2 and DW3) have interest in storing data that have associating tags to them. In this example, the administrator user should associate the tag Person:Professor to DW1, the tag Person:Student to DW3 and the tag Person to DW2. In this way, all data that is related to the tag Person:Professor will be sent to DW1, that is, Name and Address columns of the Professor table. All data that is related to the tag Person:Student will be sent to DW3, that is, Name and Email columns of the table Student. All data that is related to the tag Person will be sent to DW2, that is, all data related to the hierarchical tags Person:Professor and Person:Student, as previously explained, and also specifically the Registry column of the Professor table and City column of Student table. It is worth noting that the DW2 will store data from two different entities. It means that, in this case, the entity resolution is needed to keep the consistency and the integrity of the data. However, this step is related to the transformation phase of the ETL process, which in turn is out of the scope of this paper. On the other hand, the hierarchical tag Person:Secretary will not be used by any data warehouse and then, although the data related to tag Person:Secretary (photo column of the Professor table) is sent to the ETL process, this data will not be stored in the target data warehouses. So, the Data Magnet will automatically identify and discard data from tag Person:Secretary.

Fig. 3 depicts the star schema of this hypothetical scenario illustrated so far. The dimension tables are Person, Professor and Student. So, one example of star schema is illustrated and represents the dimension tables Student and Professor and the fact table Course. All these tables of dimension are associated to the table of fact Course by the PK attribute.

Fig. 4 illustrates the architecture of the Data Magnet. This figure represents the whole conceptual architecture of the Data Magnet and its components, as well as its workflow. The architecture of the Data Magnet will be explained in the next sections.

### *3.1. Repositories of configurations*

The settings of the Data Magnet should be stored into repositories permanently so that the Data Magnet can access it whenever needed to get all the configuration values. The required configuration parameters are: 1) data of interest and tags; 2) tags; 3) target data warehouses; 4) the connection metadata of the target data warehouses; 5) data warehouses and their respective tags. In Fig. 4 all repositories that store the configurations of the Data Magnet are represented by the Repository component in the Management Environment. In particular, all configurations must carried out at least one time by means of the administrator. After that, the Data Magnet can access all the repositories whenever needed. This feature enables the Data Magnet to work in a non intrusive manner and then aid to reaching the goals of the Data Magnet.

To be able to store all configurations, the Data Magnet makes use of the following repositories:

- **Metadata**: this repository stores all connection metadata of all target sources that is used as a data warehouse. It is needed to allow the Data Magnet to connect to the target data warehouse whenever new data is available to be stored in data warehouses. The metadata are database name, username, password, port, server name, driver name and entity name that will store data.
- **Tags**: this repository manages the tag names that is assigned by means of the administrator. Besides, tags can be defined hierarchically based on a given parent tag. For example, from the parent tag Person, we can define the hierarchical tags Person:Professor and Person:Student.
- **Sources**: this repository keeps the relation between a tag and a data of interest stored in the operational data sources. For example, the relation between the name column of the Professor table and the Person:Professor tag in the example described in Fig. 2.
- **Interests**: this repository keeps the relation between a tag and the data warehouses that have interest to store data that is related to a given tag from an operational data source. For example, the relation between DW1 and the Person:Professor tag in the example described in Fig. 2.

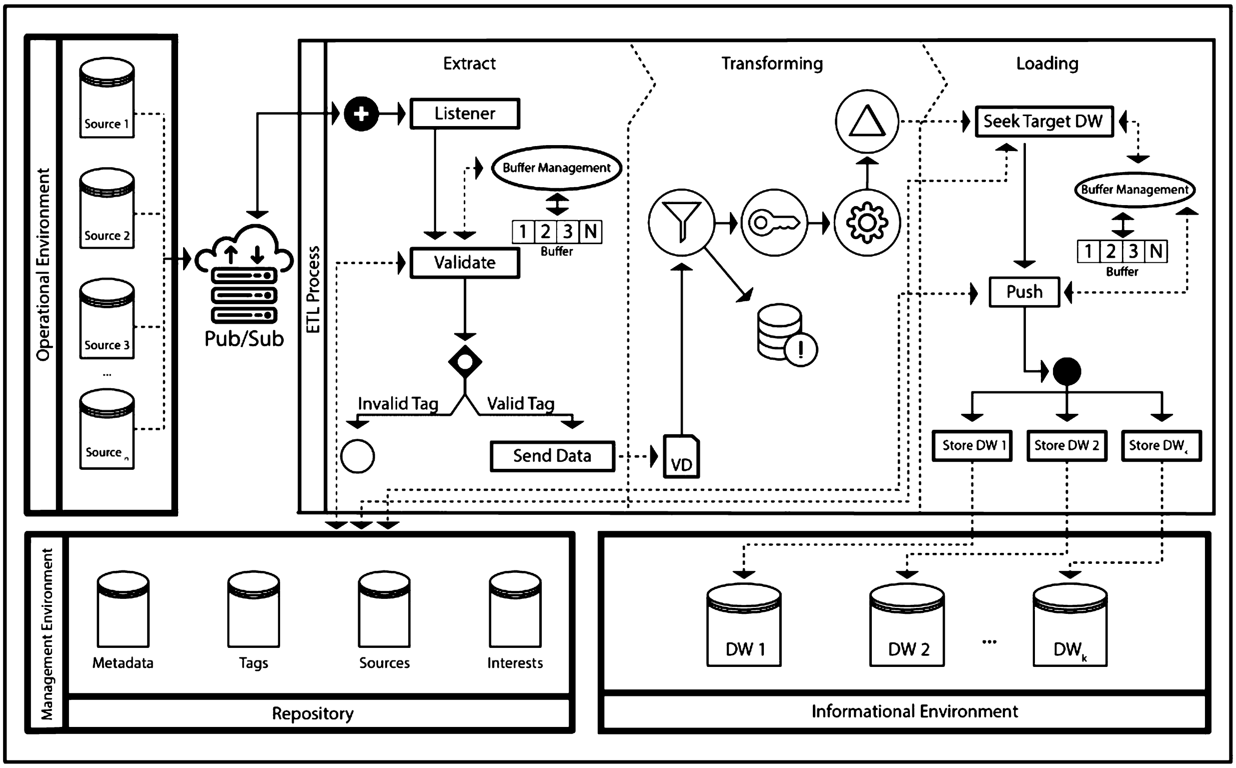


**Fig. 4.** Data Magnet Architecture.

### 3.2. Components of the Data Magnet

Below, the other components of the Data Magnet architecture are described as follows:

- **Operational Environment**: This component represents a set of operational data sources. More specifically, it represents a group of data sources responsible to produce data of interest. Furthermore, it represents the component that work as a publisher, which in turn send data to the broker to be gathered by the Data Magnet.
  It is important to say that operational data sources act in an autonomous fashion and request to deliver data to the ETL process at the most appropriate time. Then, the operational side can be adjusted to a given real time requirement. Thus, the data sources in general choose to deliver information immediately to the DM and need to guarantee that the datasets will be managed in the DW immediately after data is produced by sources. Then, sources become important components of the DM to comply with real-time requirements. More specifically, the operational data sources themselves define how is guaranteed the property of real-time, due to reactive property and non-intrusive property of the Data Magnet.
- **Pub/Sub**: the main goal of the component is to work as a publish/subscribe system, which in turn receives the data of interest and makes the data available to be consumed by the ETL process. Moreover, this component represents just a way to enable the communication and sharing of data between two heterogeneous environments. Therefore, this component is used to integrate the whole Data Magnet architecture.
- **ETL Process**: this component is regarding to the ETL process and its extraction, loading and transformation phases. The ETL process was designed over the parallelism idea. More specifically, whenever a new data is sent to the ETL process from a publish-subscribe system, then, the Listener gets the dataset and delivers the dataset to the remaining components of the ETL workflow. This allows ETL workflows to deal with its tasks with a huge volume of data received in a parallel fashion.
  So, the following components can work in parallel with a huge volume of data.
  - **Extraction**: it is responsible for receiving datasets from sources. It is composed of the Listener.
    - **Listener**: the component is responsible for connects to the broker and awaits for the data sources from operational environments send new data. Once data notification is gathered, the Listener gather the datasets and stores the datasets temporarily in the list named New Data (ND).
      The DM should get datasets in a certain standard structure. More specifically, sources must deliver data in a given structure that can be recognized, accepted and accessed by DM. The standard data organization is: (data name, value of data, type of data, list of tags, name of the operational data source, timestamp). From this component, all the ETL process starts parallelly. From the name of the data, value of data and type of data, the Data Magnet can detect the data itself and its respective type and value. Also, the Data Magnet can detect which tags are associated with the data and its operational data source.

    From all these values of metadata, the Data Magnet can detect which data warehouses have interest in storing this data that already has its type, its value and where it should be stored. As we can note, by getting these requested metadata using a standard format, the Data Magnet can deal with the schema resolution implicitly, which in turn is required in all database projects of heterogeneous environments.
    - **Validate**:once data is received, this component works to validate the data that was sent by the data sources from the operational environment. This process checks if the dataset has tags that are valid. Another validation is to check whether the received data is using the correct format to send data to the Data Magnet. When the validation is false, then, the ETL is aborted, and an error is reported to the operational data source. Otherwise, the dataset is delivered to a component called Send Data. This process is carried out with support of the repository of Tags managed by the Repository component.
    - **Buffer Management**: it is the component that stores data sent from the Validate process after the data is validated. The main interest is on tags. Once stored into this buffer, the next time that this data is received from the Listener, it is checked whether this data is already into the buffer. If it is true, the Data Magnet does not access the Repository component to validate tags. Thus, we can reduce the access to a repository and consequently obtain a great performance improvement.
    - **Send Data**: This component delivers validated data to the Validated Data (VD) component of the transformation phase of the ETL workflow. The dataset sent to this component is sent from the Validate process or by the Buffer Management.
  - **Transformation**: it is responsible to carry out the transformation phase. However, it is worth to highlight that in the Fig. 4, the representation of the transformation phase is just to exemplify the phase, as this phase is out of the scope of this paper. So, once the transformation phase is completed by a separated workflow of tasks, the data is sent to the Loading phase of the ETL process.
    - **VD**: it is a list that represents the storing of data sent by the Send Data component. The data stored into this list is maintained temporarily before being used by the transformation phase of the ETL workflow.
  - **Loading**: it represents the loading of data in the ETL workflow, that is, loading validated and transformed data into the target data warehouses. As soon as the data is sent from the transformation phase to the loading phase, this data is received from the Seek Target DW component.
    - **Seek Target DW**: it is a component that receives data from the Send Data component and immediately looks for a data warehouse who is interest in storing the data. This process is performed with the support of the repository of Interests into the Repository component. Once identified the data warehouses, the data is sent to a Push component. Also, this component stores the relation between the data of interest, the tag, and the data warehouse into a Buffer.
    - **Buffer Management**: this component is responsible for managing the Buffer component. It temporarily stores the relation between the data of interest, the tag, the data warehouse, and also the metadata for the connection with the data warehouse. Once stored into the buffer, the next time that this data is received from the Send Data component, it is checked whether this data is already into the buffer. If it is true, the DM does not connect to the Repository component to obtain all DW that require store this dataset nor to access the Repository component to obtain the metadata of the connection for the data warehouse. Thus, we can reduce the access to a repository and consequently obtain a great performance improvement.
    - **Push**: this component receives data from the Seek Target DW component. After that, it should obtain connection metadata from the identified data warehouse. If the connection metadata is not stored into the buffer, this process is performed with support of the repository of Metadata into the Repository component. Otherwise, it is performed by accessing the buffer.
- **Informational Environment**: this component represents the data warehousing environment that maintains all the target sources that represent the data warehouses interested in storing data from the operational data sources.

After we showed all properties that the Data Magnet was designed and its workflow, we can note that all these features into the Data Magnet are needed to guarantee its availability and scalability, and the feasibility of the whole ETL process. By scalability, we consider the following aspect: the increase of the number of sensors and the increase of the frequency in which the data generation occurs. Both scalability of sensors and frequency increases the data volume manipulated by the Data Magnet. The Data Magnet is scalable, that is, it continues performing its tasks maintaining an appropriate performance, as is shown in section 4. Scalability is reached mainly by the adoption of the following properties: non-intrusive, reactive, publish/subscribe system, buffer management and decoupling between operational environment and ETL process. All these features can also aid to reach the availability property, as it performs its tasks without a strong coupling between operation data sources and the ETL process. The deployment of the Data Magnet in a cloud server also could aid to reach the availability property.

## 4. Experimental evaluation of the Data Magnet

We present here the experimental evaluation performed to assess if the DM is able to support real time ETL workflows. In this sense, we analyzed the response time, availability, and scalability of the Data Magnet. The experimental tests were performed in the smart farm domain, more specifically in a dairy farm domain. The choice of this domain is because it has the expected characteristics to allow gathering data using real-time ETL processes. The tests used real and synthetic data in the context of a dairy farm.

This section has the following organization. Section 4.1 describes the motivational example that was considered for all experimental tests. Section 4.2 starts the experiments section highlighting the measures. Section 4.3 depicts the experimental setup and compare the results obtained using experimental tests with real data. Section 4.4 presents the experimental setup and compare the results obtained using experimental tests with synthetic data.

**Table 2**
Decreasing the milk production for a cow.

| Date | Cow | Milking 1st | Milking 2nd | Milking 3rd |
|---|---|---|---|---|
| 01/01/2020 | 1 | 30 | 23 | 15 |
| 02/01/2020 | 1 | 30 | 23 | 17 |
| 03/01/2020 | 1 | 28 | 22 | 18 |
| 04/01/2020 | 1 | 27 | 21 | 16 |
| 05/01/2020 | 1 | 26 | 10 | 4 |

### *4.1. Motivational example*

In our smart dairy farm, the collection of milk produced by cows is monitored by sensors and takes place three times: in the morning, at noon and in the afternoon. There are around three hundred cows in milk production every time. Moreover, it is possible to collect milk for up to thirty cows simultaneously in the milking parlor. However, currently there are bigger dairy farms in the world that can reach around six hundred cows in milk production each time and collect milk up to sixty cows simultaneously in the milking parlor. In all cases, each cow has a required milk production deadline usually from five to ten minutes. This deadline is required because each cow has biological particularities in milk production, that is, the cow has its own time to make available its milk production. Also, all cows that are inside of the milking parlor at the same moment have another deadline, that is, after a group of cows has finished the lactation process simultaneously, all cows leave the milking parlor to free up space for another group of cows. The cows that had just to leave can wait up to ten minutes to definitively leave the milking parlor. This deadline is important to the dairy farmer because they can make any decision-making immediately or as fast as possible if cows present some anomaly, as the cows are still near them inside the milking parlor.

Each volume of milk produced by a cow is important to allow computing the total volume of milk produced by the cow herd by time, by day, by week, by month, by year and so on. Furthermore, this data is important to analyze the biological behavior of each animal. In other words, the farm's financial income is directly dependent on milk production. As the cow produces more milk, the milk volume produced by the cow herd will increase and in turn also will increase the farm financial income.

Beyond the milk volume generated by cows, the milk quality is another important characteristic for the farm financial income. Daily, the quality of milk is assessed to indicate how suitable is its hygiene and biological conditions to be able to follow its industrialization and commercialization. In some cases, when the quality of milk shows some contamination or an unsuitable quality as expected, all milk volume produced at that time is discarded. It means that the more milk is produced, the care must be even greater in order to produce milk with good quality. The lack of care can cause the loss of all milk volume and big loss of money. Therefore, the farm financial income is directly affected by the lack of care with the cows and in turn due to the low quality of the milk produced.

To make it possible for the cows to produce even more milk, there are a lot of aspects to be analyzed, such as: genetic pattern, feeding, care with the cows and the environment where the cows live. In fact, the more comfortable the place where cows live, the less they will suffer from stress and the more will be their milk production. However, some factors such as food intoxication, mistreatment, attack of venomous animals and even the animal heat can cause, by a given period, unexpected and decreasing of the milk volume production.

By default, there is a natural and ordinary decrease in the milk production by every cow in each time from morning to afternoon. However, an unexpected and huge decrease in the volume of milk produced by a cow is not acceptable and it should be investigated as soon as possible. For instance, a given cow may have suffered a food intoxication and consequently its volume of milk produced has immediately decreased. Thus, this issue should be analyzed and solved as soon as possible, under the risk of infecting other cows. It is worth noting that in general there is no productivity pattern to characterize a loss for a given cow. The pattern to be considered is always based on historical milk production of each cow individually. In other words, as the lactation days go on, the animal will show its value of milk production that will be considered as regular.

Table 2 shows an example for a given cow and its lactation in different times. During January 1st and 4th, the cow kept on stable production, that is, the oscillation on its milk production can be considered as acceptable. However, on January 5th, the animal had an unexpected decrease in the milk production, specifically in its milk production between first and second milking. Moreover, the third milking production showed another decrease even higher. In this case, the dairy farmer (data analyst and decision maker) should analyze the cow behavior immediately, as well as analyze which potential factors took place to cause the unexpected decrease in the milk production.

However, in general an unexpected decrease of the milk production is noted and analyzed by the dairy farmers only some days after the fact has occurred. This situation occurs because the dairyman that act directly with the cow herd, although they learned how to deal with some situations, sometimes they do not note some anomalies with animals or just they think this is a daily normal situation. For example: after two days it is noted that a cow suffered a snake bite or a food intoxication. Consequently, it is noted that its milk production has greatly decreased. This fact negatively and directly implicates in the low milk production of the cow as well as can affect its milk quality and consequently affects the commercialization of the milk production. Moreover, every day without producing a suitable milk volume, it can affect the farm financial income.

Another important factor in dairy farms is the daily management of the volume of milk produced by the cow herd. In this sense, the dairy farmers take notes in papers, in milking control systems or spreadsheets, whose main goal is to register the amount of milk that was produced by each cow in each time. After a certain time, the dairy farmers should get historical data of each cow to be able to make decision-making analyses over this data. In cases that detected some decrease of milk production, they identify the time in which the decrease occurred and then can identify what could happen with the cow at that time. However, the dairy farmers do not have an accurate, real-time, and reliable control of the volume of milk produced in the farm, because this data is collected by third parties. Furthermore, even with an accurate value available on milk production, it is a heavy task to analyze the historical data, make decision-making analyses over the data available in papers, spreadsheets or even using a control system that does not cope with historical and real-time data.

From that context, we can note that decision-making analysis according to volume of milk produced, as well as the detection of anomalies that occurred with cows are not performed in real-time, nor immediately after some anomaly occurs, nor immediately after the required deadline ends and in general nor at the same day that the anomalies occurred. This situation happens because of a bad quality of collect and management of milk production data, lack of accurate and reliable data and lack of fast answer to the milk production analyses. Moreover, we can note that historical data assumes an essential role in the milk production analysis process. It occurs because the older data of milk production for each cow serves as a base to characterize the behavior of the cows, to make predictions over the milk production and to identify external factors responsible for the decrease of the milk production.

In our smart farm domain that uses several sensors to monitor the milk production of the cow herd, we were able to perform an experimental evaluation of the Data Magnet. Analyzing the environment in which the case study was built, we can note two important factors: 1) from the historical data point of view, it is essential to make use of a data warehouse to store the milk production data. As we mentioned before, the data warehouse can perfectly cope with a large volume and historical data. Thus, the data warehouse is appropriate to attend the dairy farm, aiming to store milk production data and in the future, whenever owners want, to get the historical data of each animal quickly. In this way, we can provide a fast, reliable and accurate value of milk historical data for each specific time; 2) from extraction in real-time and data analysis point of view, the usage of a sensor is important to automatically allow the data extraction related to the milk production, such as the volume of the milk. Thereby, it is possible to make use of DM to delivery all data produced by sensors and store them in the target DW accomplishing the real-time requirement. After that, it is possible to carry out OLAP queries using data stored in DW and make available the answers to the questions from the owners.

Before we show the experiment results, it is important to highlight two important factors: 1) from the real-time perspective, we target the soft real-time. Also, we will accomplish the deadline imposed by the smart farm domain. This scenario clearly shows what we discussed in section 1.2. Even though the dairy farm does not produce a huge data volume, there is a need to make the data of milk available in real time in the DW. In other words, there are some kinds of domains that do not produce a huge data volume in high frequency, but even so also need to attend real-time requirements (hard realtime or soft real-time). Thereby, the soft real time requirements and just respecting the deadline requirements imposed by the smart farm domain are suitable to apply the Data Magnet; 2) we did not find a dairy farm that has all equipment and sensors to make the experiments feasible. Moreover, some dairy farms had some equipment, but they are third-party equipment, and we could not access them and not even access the data produced by this equipment. For this reason, we had to build our own IoT environment composed of sensors, microcontrollers, a publish/subscribe system, and the Data Magnet itself to make it feasible to validate our proposal.

### *4.2. Experiments*

Response time (or elapsed time) is an effective manner to assess the time waste by DM. But this measure is insufficient to show the whole performance of DM as increasing the number of sensors or as the frequency of the datasets generation increases. Other derivative measures must be considered to produce better analyses of the results. Such derivative measures for response time are arithmetic average, median, percentile (p95 and p99), minimum, maximum, variance and standard deviation. In fact, it is necessary to additionally use the median and percentile measures. For this purpose, we kept a list of the performed operations, which were sorted in descendant order, that is, the fastest to the slowest response time.

Median, also called the percentile p50, is the value of time that is positioned in the middle of the set of operations. It is possible to improve the analysis, by analyzing percentile, to see the performance of a given operation in an adequate response time. We consider the percentile that corresponds to 95% of the operations (p95) and 99% of the operations (p99). The minimum response time is core to highlight the best performance to carry out all the operations. On the other hand, the maximum response time is crucial to highlight the worst performance to carry out all the operations.

The standard deviation indicates the dispersion degree of a given dataset. In fact, this measure represents the homogeneity of the results. As near is the value from zero, the results are more and more homogeneous. For this experiment, we use seconds. In turn, variance indicates how far a data set is from the average. So, a low variance indicates that all results are near average. Otherwise, a high variance indicates that the results are far from average.

We compare the DM against traditional triggers. The trigger technique is a commonly used solution to carry out the extraction phase. More specifically, we will compare the performance results and also the scalability property of the DM against triggers. For this purpose, we built a system prototype to produce milk data with similar characteristics of real data.

### 4.3. Experiments with real data

We describe in this Section the experiments based on a real case study. The Section 4.3.1 shows the experimental setup and Section 4.3.2 describes the results.

As previously pointed out in Section 3, to support real time in a non-intrusive way and reactive way, the Data Magnet should first be configured. These settings include: 1) data from sources and tags; 2) tags; 3) target data warehouses; 4) the metadata of connection for the target data warehouses; 5) data warehouses and their respective tags. After that, DM is ready to carry out its tasks in a reactive fashion and non-intrusive way.

Data of interest and tags were provided by ESP32 microcontrollers in JSON files. We set one tag named Milk and assigned it to four fields from the JSON file. These fields represent milk volume produced by the cow, the number of cow earring, date, and time. The settings were stored respectively into Tags and Sources repositories. To meet the data warehouse requirement, we defined a cloud MySQL database as a data warehouse. Also, we stored all connection metadata from it into the Metadata repository. Further, we set the tag Milk to data warehouse into the Interest repository. It is important to highlight that all these settings were stored into its respective repository in the Management Environment, as we described in section 3.

#### 4.3.1. Workbench

The dairy farm environment was based on the motivational example described in Section 4.1, which was used to carry out experiments with real data. The equipments that were used to build the dairy farm environment: three microcontrollers ESP32 LoRa Healtec, one short range presence sensor Viaonda (up to sixty centimeters), one middle range presence sensor (up to two point five meters), three earing RFID EPC-GEN2 and one flow sensor YF-S403. The Internet signal was a shared home with 10 MB and two routers TP-Link TL-WR840N. The available software were the following: the Data Magnet itself, the cloud MySQL database for the data warehouse, the local MySQL database for the metadata repository and the Mosquitto for the publish subscribe software. The DM was installed into a MacBook Pro using MacOS 10.15.7, processor 1.4 Intel Core i5, and 8 gigabytes of main memory.

The ESP32 LoRa is a microcontroller responsible for managing data that are generated by sensors. It means that each sensor is connected to an ESP32. In other words, a sensor should be connected to an ESP32 so that the sensor can generate data and ESP32 can manage the generated data. This management by microcontrollers includes getting data from sensors, storing them temporarily and sending them to a specific target.

The short-range presence sensor was responsible to detect the number of earring that is plugged in the cow's ear when a cow is inside the milking parlor. To exactly simulate the environment of the analyzed dairy farm, it was necessary sixteen short range presence sensors and sixteen ESP32. This is because the milking parlor can get data from sixteen cows in a row. However, due to the high cost of sensors and microcontrollers, the experiment was performed with just two short range presence sensors and three ESP32. So, it was possible to measure the volume of milk data of two cows in a row. Regarding the middle range presence sensor, it was responsible to detect the exact moment that a cow leaves from the milking parlor and so, the volume of milk data can be managed. Further, we used the flow sensor to collect milk data volume from cows. We collected 463 registers from the volume of milk produced by cows. These data were collected throughout 14 days, at morning period (1) and afternoon period (2) by using three cows.

The RFID earring was placed on the cow's ear. It is necessary around three hundred RFID earring to exactly simulate the environment of the analyzed dairy farm. However, we used just three RFID earring, that is, one RFID earring to each analyzed cow. So, we can identify each cow by using its number of RFID earring detected by a short range and middle range presence sensors.

Lastly, we consider the following scenario based on all sensors and microcontrollers applied in this experiment: we used three RFID earring, one to each cow. Also, we used two sets of short-range presence sensors together with an ESP32 and a flow sensor (we call Slave Set), one set of middle range presence sensors together with ESP32 (we call Master Set) and a ESP32 in a stand-alone way (we call Sender). The first set is responsible for managing milk data to two cows in a row. The short-range presence sensor detects the entrance of a cow inside the milking parlor. After that, the milk volume is collected by the flow sensor and sent to ESP32. The second set is responsible for detecting when a cow leaves the milking parlor. The number of cow earring is sent to ESP32 and all data (date, time, milk volume, number of earring) are managed. The Sender is responsible for sending data to the publish/subscribe system. The following images of the analyzed dairy farm illustrate the depicted scenario.

Fig. 5 shows the slave set. Into the milking parlor has a gallon of milk, which is responsible to temporarily maintain the milk that is collected from the cow. The short-range presence sensor is placed on a strategic place to be able to read the RFID earring placed on the cow's ear. Also, it is connected to ESP32 by means of logical ports of ESP32 itself. The flow sensor is composed of two parts: one part is the input, and the other part is output. The input part is connected to the cow's liner, whereas the output part is connected to the milk gallon. Also, the flow sensor is connected to the ESP32 by means of logical ports of ESP32 itself. By doing this assembly correctly, the ESP32 can manage all data generated by the flow sensor and the short-range presence sensor. These managed tasks are gathering data (i.e., milk volume produced by the cow and estimated by the flow sensor, date, and time), and the number of the RFID earring detected by the short-range presence sensor.

Fig. 6 shows the master set. The middle range presence sensor is placed on a strategic place to be able to read the RFID earring placed on the cow's ear when the cow is leaving the stall. Also, it is connected to ESP32 by means of logical ports of ESP32 itself. By doing this assembly correctly, the ESP32 can manage the data generated by a middle range presence sensor. These managed tasks are getting the number of the RFID earring detected by the middle range presence sensor and sending it to the slave set. It is necessary to do it just to notify the slave set that the cow that was there now has just left the stall. So, the slave set, mainly the ESP32 of the slave set, should clear all data stored into it properly to be prepared to manage data from another cow.

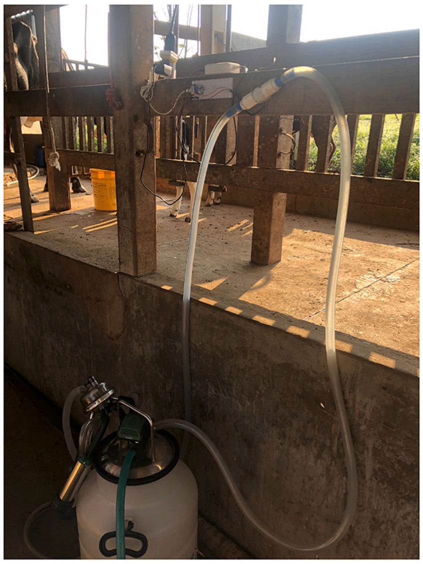


**Fig. 5.** Set of ESP32 Slave + Short range presence sensor.

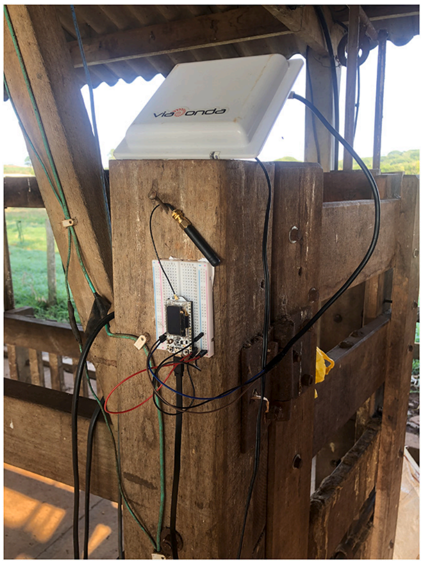


**Fig. 6.** Set of ESP32 Slave + Middle range presence sensor.

Fig. 7 shows how we assemble the Sender. This ESP32 receives data from the slave set, connecting to the Internet, and then sending this data to the publish/subscribe system. Also, it is placed on a strategic place to be able to communicate to slave sets to receive data and to the Internet to send these data to the publish/subscribe system. It is important to highlight that Sender keeps itself connected to the Internet all the time. So, the environment needs to have a permanent Internet signal.

Clearly, at this moment, we have an IoT (Internet of Things) environment composed of the Internet, sensors, microcontrollers, a sensor LAN, and all of them establish communication to each other. So, from the publish/subscribe system point of view, this environment is considered the publish side. In other words, all devices presented until now generate data and one device (Sender), which sends data (publish data) to the publish/subscribe system.

### *4.3.2. Analysis of the results*

Running this experiment using real data allowed us to assess the feasibility of the DM in a dairy farm in the smart farm domain. Also, we will analyze the execution of the Data Magnet when it is running with data generated by sensors and managed by micro-controllers. Moreover, we could analyze one of the main important requirements from real-time ETL processes, that is, the response time.

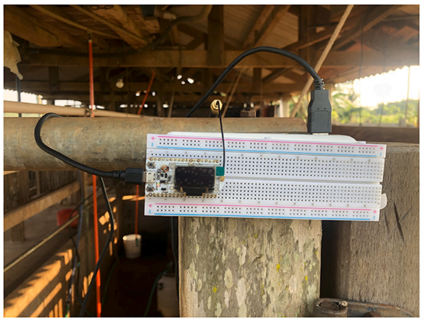


**Fig. 7.** ESP32 Sender which is responsible for sending data to the *broker*.

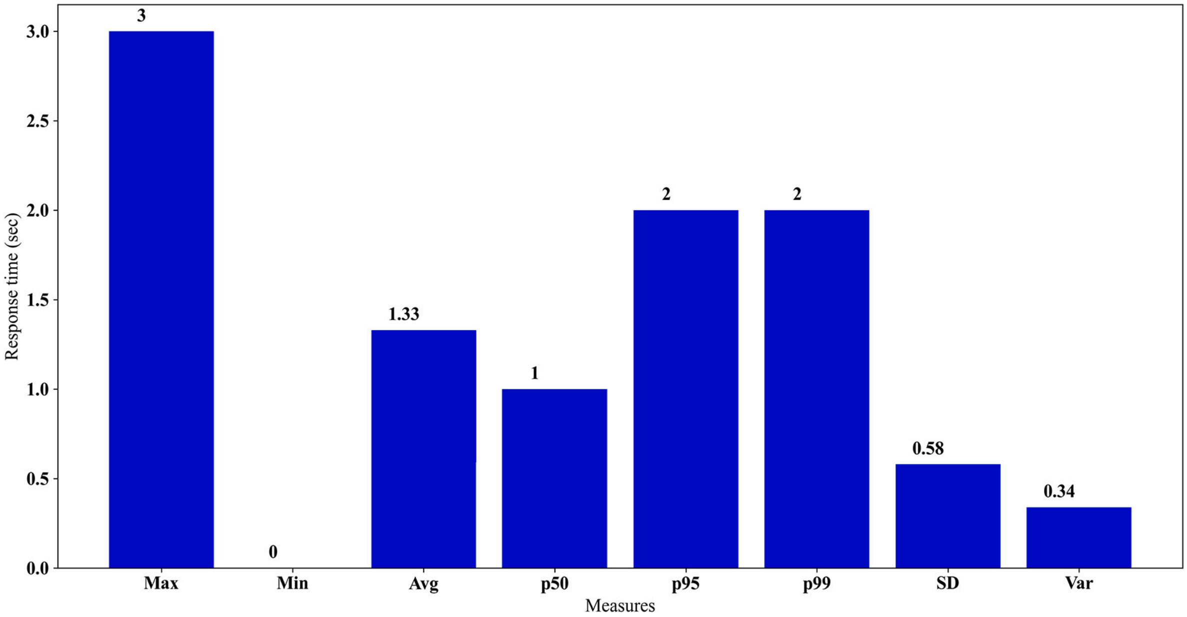


**Fig. 8.** Performance of the Data Magnet with real data.

We focused on two main analyses: 1) from the Data Magnet point of view, how good Data Magnet performed all operations according to the elapsed time? 2) Which analyses upon the collected data can be performed in real-time by applying the Data Magnet using the dairy farmer viewpoint?

These questions can be answered by analyzing two fields into the JSON file created to maintain the data generated by the sensors. The first one is milk_time, which represents the exact time that Sender sends the JSON file with all data of milk production to the publish/subscribe system. This time is obtained by the Sender itself connected to the Internet. The second one is time_arrive, which represents the exact time that the Data Magnet stored data in DW. This time is obtained by DM itself connected to the Internet. So, from these two fields, we can analyze how long the Data Magnet took to store data from the IoT environment to a target data warehouse. In other words, we can show the non-intrusive, reactive and tag properties being running properly in a real case.

Fig. 8 shows the performance of the Data Magnet. This figure presents the values of maximum value (Max), minimum value (Min), median p(50), average (AVG), p(95), p(99), standard deviation (SD) and variance (Var). All these times were measured in seconds. The value of p(50) was 1 second, whereas the value of p(95) and p(99) both were 2 seconds. The value of average was 1.33 seconds, that is, the DM took 1.33 seconds on average to perform its task based on all operations. The value of maximum time was 3 seconds, and the value of minimum time was near 0 seconds. In other words, in the best case, the DM performed in less than 1 second, and in the worst case, the DM performed in 3 seconds. The value of standard deviation was 0.58 seconds, and it means that the elapsed time was very homogeneous considering all operations. The variance was 0.34 seconds, and it means that the response time suffered low variations regarding the average and remained stable regarding all operations.

We also can notice that the DM performed in a suitable response time to the real case study. Also, we could demonstrate that tag property can make the Data Magnet act in a non-intrusive way, that is, the Data Magnet did not access the sensors or microcontrollers to get milk data. Also, the Data Magnet did not need to know how to access these heterogeneous data sources nor how to manage the data generated from them. Furthermore, by using reactive property, the DM made use of a publish subscribe software to receive milk datasets that was generated in a heterogeneous environment and automatically perform all its tasks quickly.

Beyond all these analyses, the Data Magnet was able to perform all its tasks and make milk data available into a data warehouse in a maximum and minimum aforementioned low response time. It means that the Data Magnet respected the deadline imposed by the smart farm domain itself (of the order of 5 up to 10 minutes) and it could make milk data available almost immediately to be stored into the data warehouse.

### 4.4. Experiments with synthetic data

Although the Data Magnet was validated using a real case study, the use of real data was not enough to assess all the requirements presented in the Section 4. This is because the volume of data generated by the real case study was small. So, the experiment with synthetic data was built to validate all the requirements and to validate the feasibility to apply the Data Magnet in other environments with the same characteristics of the real case study. To better understand the environment of this experiment, we consider the same motivational example depicted in Section 4.1 and the same measures depicted in Section 4.2.

It is important to note that the scenario used to gather synthetic data was similar to the scenario of the real case study. Therefore, the data generated synthetically by the Synthetic Data Generator (SDG) uses the same characteristics of data produced by sensors in the real case study. More specifically, data were produced in JSON files, providing fields that indicates the number of cow earring, date, time, and milk volume generated by a cow and its value has the same semantic of real data. So, we defined default values to these fields, which represent the same range of values gathered into real case study. Consequently, we kept similar characteristics of the real data and their values. Besides, we controlled the frequency and data volume that milk data was generated. As a consequence, we are able to analyze the execution of DM and then we can evaluate the requirements: elapsed time, availability, and scalability.

Section 4.4.1 describes the experimental setup. Section 4.4.2 analyses the influence of the increase of the number of sensors delivering data to the publish-subscribe system. Section 4.4.3 analyses the influence of increasing the frequency in data generation.

#### 4.4.1. Workbench

To carry out experiments using synthetic data and allow us to compare the Data Magnet to another solution, we applied the same experiments to both our proposal of the Data Magnet and the most used technique to solve the real time ETL workflows using triggers.

For DM, we used the following software: 1) DM, which run in a MacBook Pro with operation system 10.15.7, 1.4 Intel Core processor i5, 8 GB of main memory; 2) SDG, which performed in a Dell 15R computer desktop, Intel Core i5 processor, 8 GB of main memory and Windows version 10 with 64 bits; 3) Cloud MySQL database to represent DW; 4) Local MySQL database to represent a metadata repository; 5) Eclipse Mosquitto to represent a publish subscribe software. The Internet was a shared home Internet of 10 MB bandwidth and 1 router.

Besides, another configuration was carried out to allow us to run the experiments for the Data Magnet. The configuration is of the Data Magnet environment. As depicted in section 4.3, the Data Magnet should be configured. So, to perform the experimental test of the Data Magnet with synthetic data, the settings are nearly the same as the experiments using real data, just the operational data sources were different. These settings include: 1) data from sources and tags; 2) tags; 3) target data warehouses; 4) the metadata of connection for target DW; 5) data warehouses and their respective tags. After that, DM was ready to carry out its tasks in a reactive way and non-intrusive fashion.

For the trigger technique, we used the following configuration: 1) SDG, executed in a Dell 15R computer, Intel Core processor i5, 8 GB of main memory and Windows version 10 with 64 bits; 2) Cloud MySQL database to represent a DW, which was the same that was used for the Data Magnet; 3) Local MySQL database to represent a database in which it stores the data generated by the SDG. Also, into this local repository we created a table that was responsible for storing the data generated by the SDG. Moreover, into this table, we created a trigger that sends the newly inserted data from the table of the repository to the ETL tool; 4) The ETL tool, a python application that we created to simulate the ETL process. This tool performs the extraction phase firstly, that is, it gathers the data that is generated by the trigger and then sends it to the data warehouse. The Internet was a shared Internet of 10 MB of bandwidth and 1 router TP-Link TL-WR840N.

For both the Data Magnet and the trigger, we should configure the SDG. To produce synthetic data, it was set the number of synthetic cows, the frequency that the milk data should be produced by each sensor, and the amount of synthetic sensors. We also consider all thirty, sixty or more cows having its milk collected by sensors simultaneously (as we highlighted in section 4.1). The generation of data was performed by the proportion of eight cows conducted for each sensor and as we double the number of cows we also double the number of sensors.

#### 4.4.2. Scalability of Data Magnet by increasing the number of synthetic sensors

This experiment was performed to assess scalability of DM. More specifically, it was carried out analyzing the elapsed time of the sensors simultaneously delivering milk data. The following configurations were used: C1) 8 synthetic sensors handle data of milk from 62 synthetic cows; C2) 16 synthetic sensors handle data of milk from 125 synthetic cows; C3) 32 synthetic sensors handle data of milk from 250 synthetic cows; C4) 63 synthetic sensors handle data of milk from 500 synthetic cows; C5) 125 synthetic sensors handle data of milk from 1,000 synthetic cows. The frequency of generation of data was 30 seconds. In fact, the increase of the number of sensors with constant frequency indirectly increases the volume of data managed by the Data Magnet.

All performed configurations agree with the current scenario of dairy farms. Nowadays, dairy farms have approximately 600 cows in lactation. In fact, we go further and set a configuration that has 125 sensors managing 1,000 cows.

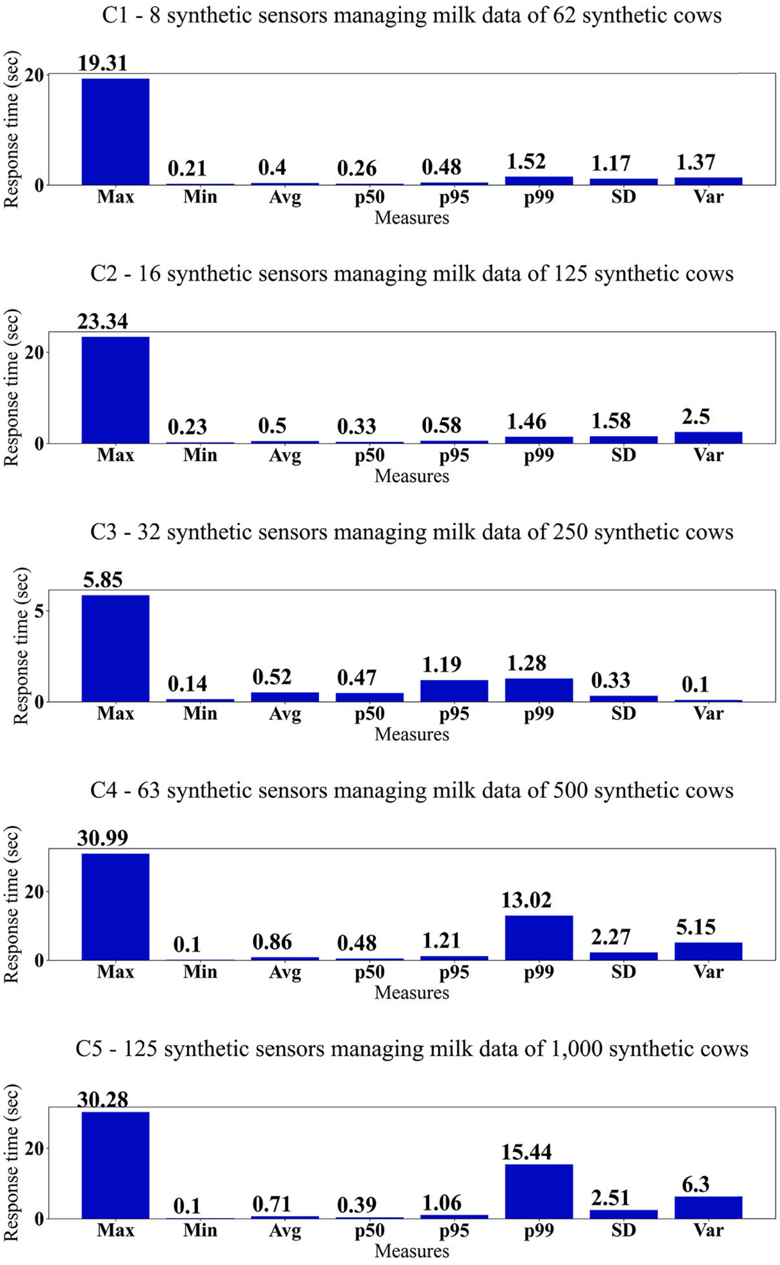


**Fig. 9.** The response time as increasing the number of sensors sending data simultaneously. C1 is related to 8 sensors managing and delivering milk data from 62 cows. C2 is related to 16 sensors managing and delivering milk data from 125 cows. C3 is related to 32 sensors managing and delivering milk data from 250 cows. C4 is related to 63 sensors managing and delivering milk data from 500 cows. C5 is related to 125 sensors managing and delivering milk data from 1,000 cows.

Fig. 9 shows the results. The Fig. 9 is divided in 5 sub-figures, each one is related to one configuration (i.e., label with the configuration number). Fig. 9-1 represents 8 sensors handling and delivering milk data from 62 cows. Fig. 9-2 represents to 16 sensors handling and delivering milk data from 125 cows. Fig. 9-3 represents to 32 sensors handling and delivering milk data from 250 cows. Fig. 9-4 represents to 63 sensors handling and delivering milk data from 500 cows. Fig. 9-5 represents to 125 sensors handling and delivering milk data from 1,000 cows.

Altogether, this experiment collected 27,668 milk data. In fact, we represent the execution of DM using the same milk volume and number of sensors of the real-case study. We also compare how efficient DM was as increasing the number of sensors. The Max value of the configurations C1, C2, C4 and C5 kept among 19.31 and 30.99 sec. In percentage, this indicates 0.57% of all results gathered to all configurations. The Max value of C3 was 5.84 sec. It is worth highlighting that in some tests we do not show the Max value as well as the explanation about it. In rare cases, the heterogeneous environment created to test those experiments can undergo instability of the whole environment as well as of the Internet sign and in turn inevitably a tiny quantity of data can reach a higher response time in relation to the rest of the collected data. For the Min value, in all configurations this value was very homogeneous as it had values among 0.1 and 0.23 second. Then, we can argue that the DM can gather data from sources and store the data in DW in less than 0.3 seconds. The maximum average time obtained was 0.86 seconds and the minimum average time was 0.40 seconds. Therefore, as increasing the number of sensors, the average of elapsed time remained very well stable. In fact, we expected to obtain a linear growth in the performance degradation, but the results were even better.

Regarding percentiles, p95 and p50 were very stable as increasing the number of sensors. p50 and p95 kept among 0.26 second and 1.22 second. Therefore, as increasing the number of sensors, the highest value of elapsed time of 95% of all data gathered from sensors was smaller than 1.3 second. For p99, its was 15.43 second and 13.01 second for C1 and C2. Also, for the C3, C4 and C5, the p99 was even better of 1.28 second, 1.46 second, and 1.52 second. The variation in the values of p99 could be explained by the implicit overhead for data exchange in on-line transaction systems. It occurs in function of the limitations of data transfer rate, network protocol overhead, hardware limitations, the use of a shared internet or other kinds of restrictions that increase the time to deliver the data to the data warehouse. In particular, the internet oscillation very likely was the main reason since the internet was a shared home internet with only 10 MB bandwidth.

Besides, DM remained very stable as increasing the number of sensors. By doing that in a parallel way, the Data Magnet performed all tasks in a suitable and desirable elapsed time. Also, some values out of pattern occurred because a very few operations, which were carried out in a given time that the shared Internet was intermittent. In fact, sometimes a variation of the Internet signal can occur and the slightly growth of the elapsed time. Fig. 9 shows these facts for the value of p99.

Fig. 10 illustrates a match of the elapsed time among DM and the triggers as increasing the number of sensors. Fig. 10 is divided in 5 sub-figures: Fig. 10-1 represents 8 sensors handling data from 62 cows, Fig. 10-2 represents 16 sensors handling data from 125 cows, Fig. 10-3 represents 32 sensors handling data from 250 cows, Fig. 10-4 represents 63 sensors handling data from 500 cows and lastly, Fig. 10-5 represents 125 sensors handling data from 1,000 cows.

Based on all results for the analysis of the influence of the scalability as the number of sensors increases, we can compare the difference of the elapsed time among DM and triggers according to the performance loss and the performance gain. For almost all configurations and almost all measures, except for some response time of Max value, DM produced a elapsed time improvement over triggers. Regarding Avg, the gain of the Data Magnet in comparison to the trigger technique varied from 987% to 238.803%. The gain of the Data Magnet for the p50 varied from 1.588% to 416.877%, for p95 varied from 1.196% to 302.040%, and for p99 varied from 104% to 21.522%. However, only for the configurations C1, C2 and C4, the Data Magnet presented a performance loss over Max value. This performance loss varied in these three configurations from 6% to 109%. The performance loss in Max value because of the overhead of the whole environment. In fact, the shared Internet was unstable sometimes during the experiment, producing high values for small quantities of synthetic cows. However, the elapsed time loss was a very small amount of all results generated in the experiments.

The Fig. 11 illustrates a comparison between the linear growth expected by this experiment and the constant growth of Data Magnet found as increasing the number of sensors. The DM produced a small change in its execution behavior and the elapsed time remained very stable. Besides, when the number of sensors doubled, we expected that the elapsed time doubled as well (i.e., at least a linear increase of the response time). But we observed great behavior and a very small increase in the response time. This is true for Avg, p50, p95 and p99. From all configurations, DM does not change the behavior, even in configuration C5, for which the number of sensors represents the worst configuration (i.e., 125 sensors delivering data and using 1,000 cows). Triggers show a higher elapsed time than DM.

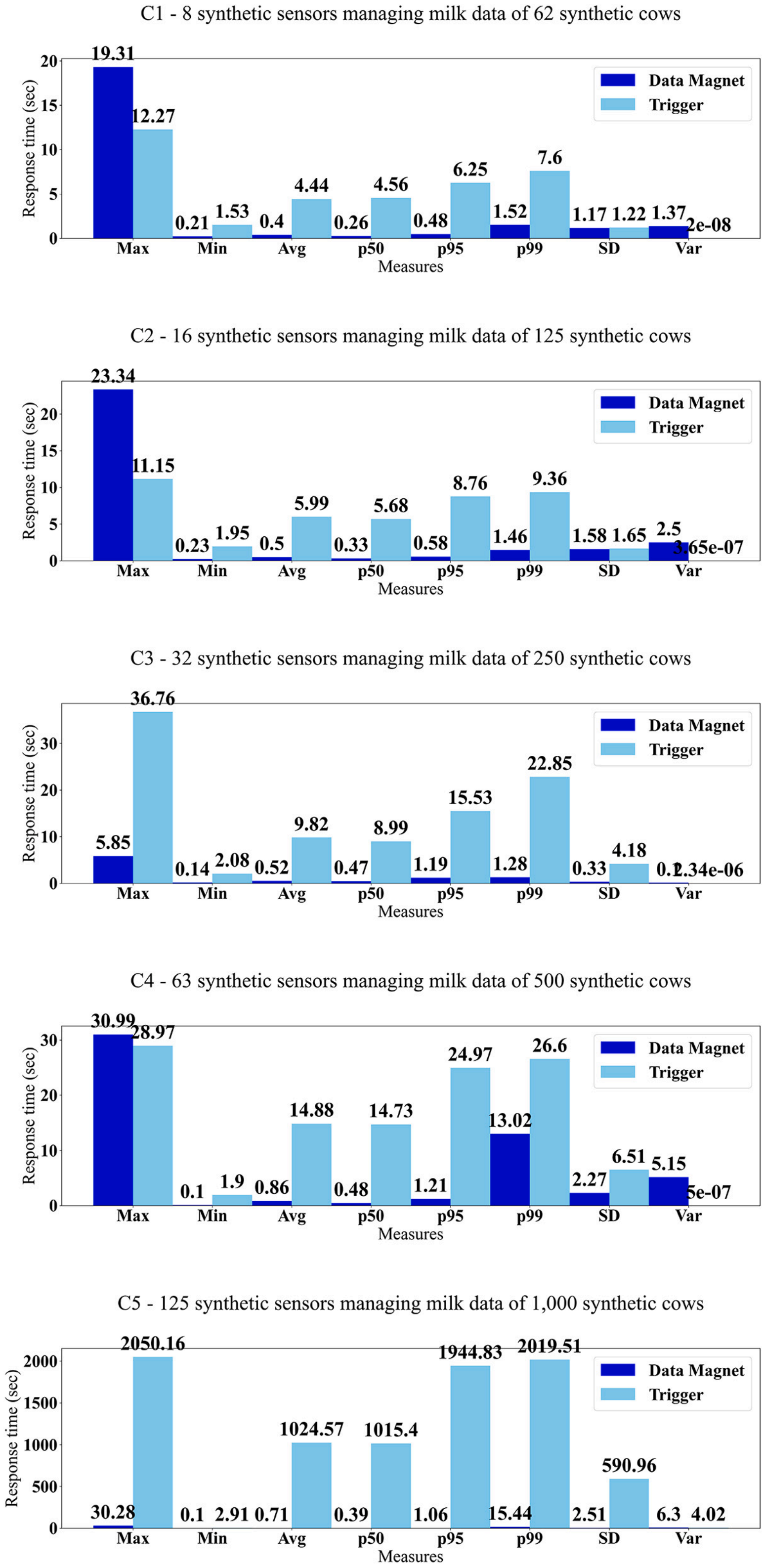
#### *4.4.3. Scalability of Data Magnet by increasing the frequency of the data generation*

Another experiment performed to assess DM analyzed the elapsed time as the frequency of the data generation growth (i.e., as the time interval to generate data decreased). For this purpose, we set five configurations as follows: for all configurations, we fixed the number of synthetic sensors managing and delivering data of milk to 63 sensors and we fixed the number of synthetic cows to 500 cows. For configuration C1, the frequency was every 120 seconds. For configuration C2, the frequency was every 60 seconds. For configuration C3, the frequency was every 45 seconds. In configuration C4, the frequency was every 30 seconds. Lastly, in configuration C5, the frequency was only every 15 seconds. In fact, the increase of the frequency of the data generation with fixed number of sensors and fixed number of cows indirectly increases the volume of data managed by the Data Magnet.

As for section 4.4.2, this experiment also reflects the current scenario of dairy farms. Currently, there are dairy farms that get milk of around 60 cows in a row by using around 8 sensors in a frequency of some minutes (5 to 10 minutes). It means that all 8 sensors can send data of milk using sources to the DW at the same time. We also performed tests at a higher frequency than the current scenario. Consequently, we analyzed the scalability of DM from another point of view, that is, as the frequency of the data generation increased.

Fig. 12 depicts the results regarding the elapsed time of DM as the frequency in data generation increased. The Fig. 12 is divided of 5 sub-figures, where each figure shows a distinct configuration. The Fig. 12-1 is related to configuration C1, the frequency was every 120 seconds, the Fig. 12-2 is related to configuration C2, the frequency was every 60 seconds, the Fig. 12-3 is related to configuration C3, the frequency was every 45 seconds, the Fig. 12-4 is related to configuration C4, the frequency was every 30 seconds, lastly, the Fig. 12-5 is related to configuration C5, the frequency was every 15 seconds.

Altogether, this experiment collected 37,494 milk data. As we can note, the value of Max for configuration C1, configuration C2, configuration C3 and configuration C5 kept between 1.75 seconds and 5.81 seconds. Just configuration C4 took 30.99 seconds for Max value. In percentage of operations, it indicates only 0.51% of all results produced in the 5 configurations. Besides, this time reached by the C4 is because of the instability of the whole environment as well as the Internet sign, as previously explained in



C1 - 8 synthetic sensors managing milk data of 62 synthetic cows

C2 - 16 synthetic sensors managing milk data of 125 synthetic cows

C3 - 32 synthetic sensors managing milk data of 250 synthetic cows

C4 - 63 synthetic sensors managing milk data of 500 synthetic cows

C5 - 125 synthetic sensors managing milk data of 1,000 synthetic cows

**Fig. 10.** Comparison between the Data Magnet and the Trigger technique. C1 is related to 8 sensors managing and delivering milk data from 62 cows. C2 is related to 16 sensors managing and delivering milk data from 125 cows. C3 is related to 32 sensors managing and delivering milk data from 250 cows. C4 is related to 63 sensors managing and delivering milk data from 500 cows. C5 is related to 125 sensors managing and delivering milk data from 1,000 cows.

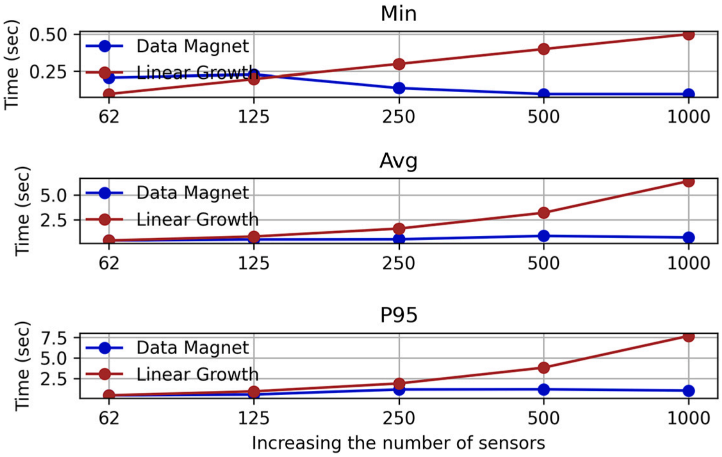


**Fig. 11.** Comparison between the Data Magnet and the expected the linear growth as the number of sensors increases. Min refers to the minimum value. Avg refers to the average value. P95 refers to the 95o percentile.

the paper. More specifically, the loss of cycles because of the Internet instability produced higher values for higher frequencies. For the Min measure, in all configurations the value remained very homogeneous. In percentage of operations, it indicates 11.6% of all results produced for the 5 configurations. For Avg, the minimum value of Avg was 0.34, whereas the maximum value obtained of the average was 0.86 seconds. Then, as increasing the frequency of the data generation, the average of elapsed time kept very stable. Therefore, the elapsed time of DM was better than linear growth.

Regarding percentile, only configuration C2 showed a value of p99 above 1.6 seconds. This is because the moment that the configuration C2 was performed the Internet signal was intermittent. It causes in a few operations a slight increase of their response time. However, the values of p50, p95 and p99 of the configuration C1, configuration C3, configuration C4, configuration C5, and the values of p95 and p50 of configuration C2 remained very stable as increasing the frequency of the data generation. The maximum value of the elapsed time of 99% of all results was smaller than 13.02 seconds, whereas the minimum value was only 0.59 second. For Var and SD, the minimum was 0.11 second, and the maximum was 2.27 second. Regarding Var, the minimum value was 0.01 seconds, and the maximum value was 5.15 seconds.

Despite the Data Magnet reached in configuration C2 30.99 seconds for the Max value and 13.02 seconds for p99, the DM remained very stable as increasing the frequency of the data generation. In fact, the Data Magnet performed all its tasks in a desirable elapsed time as increasing the frequency of the data generation. Also, these outlier values in the configuration C2 are due to a very few operations, which were carried out in a time that the shared Internet was very intermittent. It causes variation of Internet signal and consequently a huge growth of the elapsed time. These conclusions are represented in Fig. 12 from the value of p99.

It is important to highlight that DM was designed by using a publish subscribe software and Internet signal to allow sharing data from heterogeneous environments. However, the experiment used a shared home Internet. It means that in certain moments the Internet was intermittent and in turn all performed operations were affected by the intermittent Internet signal. Even so, we can reach a great and suitable response time and DM processed the tasks in a desirable time.

Fig. 13 illustrates the response time for DM and triggers as the frequency of the data generation increased. The Fig. 13 has 5 sub-figures: Fig. 13-1 is related to configuration C1, the frequency was every 120 seconds, Fig. 13-2 is related to configuration C2, the frequency was every 60 seconds, Fig. 13-3 is related to configuration C3, the frequency was every 45 seconds, Fig. 13-4 is related to configuration C4, the frequency was every 30 seconds and lastly, Fig. 13-5 is related to configuration C5, the frequency was every 15 seconds.

Based on all results showed in the Fig. 10 for analyzing the influence of the increase of the frequency of the data generation, we can analyze the difference of the response time between the Data Magnet and the trigger technique according to performance loss and performance gain. For almost all measures and all configurations, DM had a high performance gain in relation to triggers. Regarding Avg, the performance gain of the Data Magnet in relation to the trigger technique varied from 1.626% to 320.756%. The gain of the Data Magnet for p50 varied from 2.924% to 295.129%, for p95 varied from 1.952% to 359.604%, for p99 varied from 104% to 310.804%. However, only for the configuration C4, the Data Magnet provided a performance loss for the Max value. This performance loss was just 6%. As we depicted in section 4.4.2, the performance loss for the Max measure was caused by the overhead produced by the environment. However, the performance loss represents only a small amount of all results produced in the experiments. In general, DM produced a huge performance gain against triggers.

The Fig. 14 shows a comparison between the expected linear growth and the constant growth of Data Magnet found as the frequency increases. As we can note, the Data Magnet showed a good performance and a stable elapsed time as increasing the frequency of the data generation. This is shown for Avg, p95, SD and Var measures in all the 5 configurations. Considering the Min value, even in configuration C5, in which the frequency of the data generation reached only 15 seconds, the Data Magnet could be performed as great and with low response time as in configuration C1, in which the frequency was much higher (every 120 seconds).

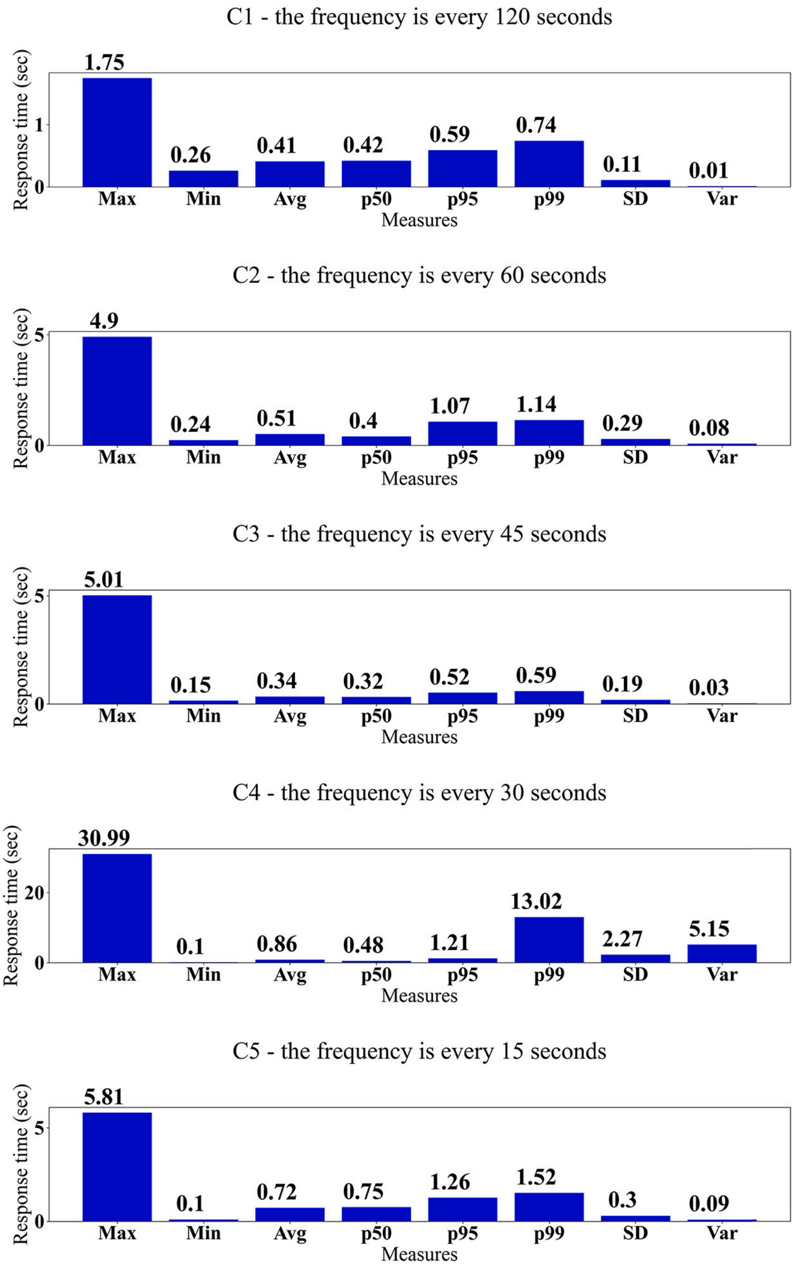


**Fig. 12.** The response time as the frequency of the data generation increases. In C1, the frequency was every 120 seconds. In C2, the frequency was every 60 seconds. In C3, the frequency was every 45 seconds. In C4, the frequency was every 30 seconds. In C5, the frequency was every 15 seconds.

The trigger technique showed a higher response time in comparison to the Data Magnet in almost all configurations. This fact can be noted in Avg, p99, SD and Var measures. Furthermore, in almost all configurations, the trigger technique did not show a response time even near to the Data Magnet, as we can see in Min, Avg, p95 and p99 measures.

## 5. Conclusions

This article presents and details the proposal of the Data Magnet (DM), an architecture to handle with real time ETL processes in data warehousing environments that is new and innovative. DM uses the following concepts: tags, reactive property and non-intrusive property. Also, as the real time requirement was not deeply discussed in related work, DN proposed an alternative way to think about the real time requirement through a soft real time perspective. This article defined the concept of real time and discussed the feasibility and applicability for systems that use real-time ETL workflows.

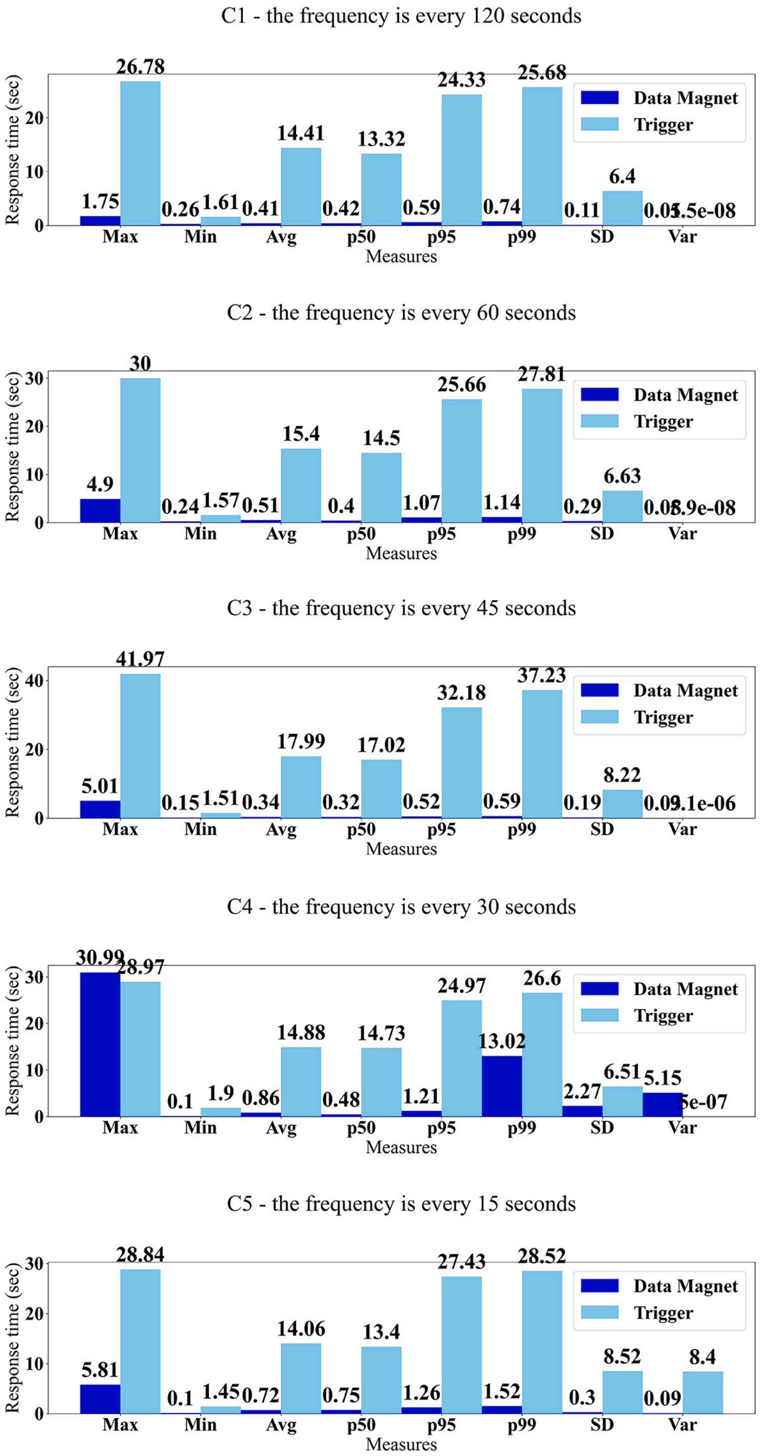


**Fig. 13.** Comparison of the Data Magnet and the Trigger technique as the frequency of the data generation increases. In C1, the frequency was every 120 seconds. In C2, the frequency was every 60 seconds. In C3, the frequency was every 45 seconds. In C4, the frequency was every 30 seconds. In C5, the frequency was every 15 seconds.

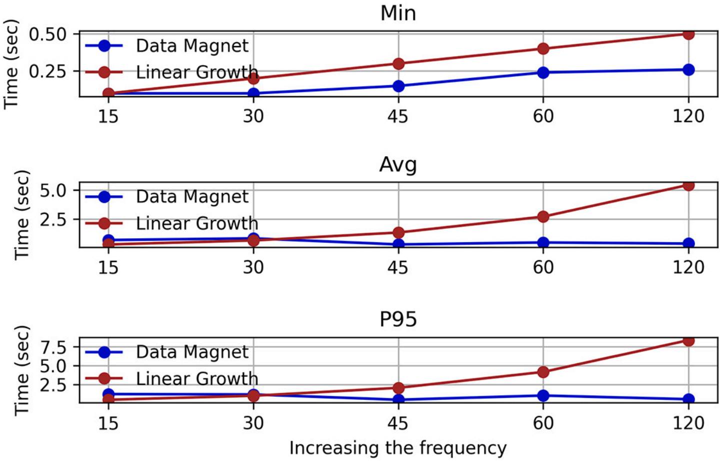


**Fig. 14.** Comparison between the Data Magnet and the expected the linear growth as the frequency increases. Min refers to the minimum value. Avg refers to the average value. P95 refers to the 95o percentile.

According to the non-intrusive characteristic, DM does not connect to sources to gather data. Therefore, the Data Magnet does not need to use wrappers, triggers, log files or delta files. Instead, DM is designed to gather datasets from operational sources. For this purpose, DM uses the Tag property. Tag is an flag that is assigned to an data item of a source and in turn it will be received and stored in a data warehouse. So, the use of tags causes a decoupling among sources and the ETL workflow. This property also allows increasing the performance of the ETL workflow, since extracting data turns less costly to be done, due to the fact that DM does not connect to sources.

Using the reactive concept, DM respond to an operation of insertion in sources and gets datasets. For this purpose, DM uses a publish subscribe software to share data among heterogeneous systems. Therefore, DM gets data items of sources, identify the data item and tags, identify the DW that have interest in handle the data assigned to the tags and store data into the target DW.

From the experiments performed by using real data, we could evaluate the feasibility in applying the Data Magnet in the smart farm domain, more specifically in a dairy farm. In this experiment, the Data Magnet provided good performance and showed feasibility to perform real-time ETL processes. Further, from the experiments performed by using synthetic data, we evaluated the requirements cited in the section 4 and the Data Magnet reached low response time, high availability and scalability. Also, the Data Magnet provided a high-performance improvement in comparison to the trigger technique.

All these characteristics of the Data Magnet represent and prove the new and innovative way to deal with real time ETL workflows for DW. In other words, it is possible to decouple the ETL process and operational data sources and make it work in a non-intrusive and reactive manner, which in turn can improve the performance of ETL workflows.

As future work, we will investigate how to improve DM by distinguishing critical and non-critical data for validation. In this sense, we intend to allow the Data Magnet to prioritize the validation of critical data and postpone the validation of non-critical data. In the current proposal of the Data Magnet there is no distinction between critical and non-critical data. We also intend to investigate the impact of the transformation phase in the real-time ETL process. Another future work is to allow the Data Magnet managing historical data and real-time data in separate repositories. Other possible future work includes the treatment of the flow of the Data Magnet in case of an error is reported from operational data sources. In this case, we should define new tasks to allow the user being aware of the error or the Data Magnet could perform some automatic tasks to manage the error.

## 6. Author contribution statement

- Flávio de Assis Vilela, Ricardo Rodrigues Ciferri: conceived and designed the experiments; performed the experiments; analyzed and interpreted the data; contributed reagents, materials, analysis tools or data; wrote the paper;
- Valéria Cesário Times, Augusto de Paula Freitas: conceived and designed the experiments; analyzed and interpreted the data, wrote the paper;
- Alberto Carlos de Campos Bernardi: analyzed and interpreted the data, wrote the paper.

## Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ricardo Rodrigues Ciferri reports financial support was provided by Federal University of Sao Carlos. Ricardo Rodrigues

Ciferri reports a relationship with Federal University of Sao Carlos that includes: board membership. Ricardo Rodrigues Ciferri has patent pending to None.

## Data availability

Data will be made available on request.

## References

[1] P. Figueiras, R. Costa, G. Guerreiro, H. Antunes, A. Rosa, 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.

[2] S. Fuentes, C. Gonzalez Viejo, B. Cullen, E. Tongson, S. Chauhan, 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 20 (05 2020), https://doi.org/10.3390/s20102975.

[3] R. Mukherjee, 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.

[4] B. Yadranjiaghdam, N. Pool, 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.

[5] 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, https://ieeexplore.ieee.org/document/8510574/.

[6] F. Sabry, E. Ali, A survey of real-time data warehouse and ETL, Int. J. Sci. Eng. Res. 5 (7) (2014), http://www.ijser.org.

[7] C.D.d.A. Ciferri, Distribuição dos dados em ambientes de data warehousing: o Sistema WebD 2W e algoritmos voltados à fragmentação horizontal dos dados, Ph.D. thesis, Universidade Federal de Pernambuco, 2002.

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

[9] M. N, 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] R. Kimball, J. Caserta, R. Kimball, J. Caserta, The data warehouse ETL toolkit: practical techniques for extracting, cleaning conforming, and delivering data, arXiv:1011.1669v3, https://doi.org/10.1017/CBO9781107415324.004, 2004.

[11] K. Kakish, T.A. Kraft, ETL evolution for real-time data warehousing, Conf. Inf. Syst. Appl. Res. 5 (2214) (2012), www.aitp-edsig.org.

[12] A. Wibowo, Problems and available solutions on the stage of extract, transform, and loading in near real-time data warehousing (a literature study), in: 2015 International Seminar on Intelligent Technology and Its Applications, ISITIA 2015 - Proceeding, 2015, pp. 345–349.

[13] N. Gajanan Mane, Near real-time data warehousing using ETL (extract, transform and load) tools, Int. J. Adv. Res. Comput. Eng. Technol. 4 (5) (2015).

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

[15] J. Liu, Real-Time Applications, 2000.

[16] M. Kleppmann, Designing Data-Intensive Applications, 2017.

[17] A. Sabtu, N.F.M. Azmi, N.N.A. Sjarif, S.A. Ismail, O.M. Yusop, H. Sarkan, S. Chuprat, The challenges of extract, transform and load (ETL) for data integration in near real-time environment, J. Theor. Appl. Inf. Technol. 95 (22) (2017) 6314–6322.

[18] X. Li, 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.

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

[20] T. Jain, R. S, S. Saluja, Refreshing datawarehouse in near real-time, Int. J. Comput. Appl. 46 (18) (2012) 975–8887, https://pdfs.semanticscholar.org/0c8c/ac4020f70b414085752f4c753e81eb8050be.pdf.

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

[22] C. Kulatunga, L. Shalloo, W. Donnelly, E. Robson, S. Ivanov, Opportunistic wireless networking for smart dairy farming, IT Prof. 19 (2) (2017) 16–23, https://doi.org/10.1109/MITP.2017.28.

[23] M.A. Zamora-Izquierdo, J. Santa, J.A. Martínez, V. Martínez, A.F. Skarmeta, Smart farming IoT platform based on edge and cloud computing, Biosyst. Eng. 177 (2019) 4–17, https://doi.org/10.1016/j.biosystemseng.2018.10.014.

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

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

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

[27] C. Thomsen, T.B. Pedersen, W. Lehner, RiTE: providing on-demand data for right-time data warehousing, ICDE 2008 (2008) 456–465.

[28] P.D.M. Plentz, Mecanismos de Previsão de Perda de Deadline para Sistemas Baseados em Threads Distribuídas Tempo Real, Ph.D. thesis, 2008.

[29] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manag. Inf. Syst. 24 (3) (2007) 45–77, https://doi.org/10.2753/MIS0742-1222240302.

[30] P. Vassiliadis, A. Simitsis, Near Real Time ETL, vol. 3, 2009.

[31] F. Penzlin, H. Schink, C. Becker, M. Krug, A. Dreiling, Current state and future challenges of real-time ETL, Computer (June 2009) 6–10.

[32] F. Farooq, S.M. Sarwar, Real-time data warehousing for business intelligence, in: Proceedings of the 8th International Conference on Frontiers of Information Technology - FIT '10, ACM Press, New York, New York, USA, 2010, pp. 1–7, http://portal.acm.org/citation.cfm?doid=1943628.1943666.

[33] D.M. Tank, A. Ganatra, Y.P. Kosta, C.K. Bhensdadia, Speeding ETL processing in data warehouses using high-performance joins for changed data capture, in: (CDC), Proceedings - 2nd International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom 2010, Cdc, 2010, pp. 365–368.

[34] R. S, S. Balaji.B, N.K. Karthikeyan, From data warehouses to streaming warehouses: a survey on the challenges for real-time data warehousing and available solutions, Int. J. Comput. Appl. 81 (2) (2013) 15–18, https://doi.org/10.5120/13984-1990.

[35] N. Ferreira, P. Martins, P. Furtado, Near real-time with traditional data warehouse architectures: factor and how to, in: Proceedings of the 17th International Database Engineering & Applications Symposium on - IDEAS '13, 2013, pp. 68–75, http://dl.acm.org/citation.cfm?doid=2513591.2513650.

[36] S. Bouaziz, A. Nabli, F. Gargouri, From traditional data warehouse to real time, intelligent systems design and applications, https://doi.org/10.1007/978-3-319-53480-0, 2017.

[37] R.M. Bruckner, B. List, J. Schiefer, Striving Towards Near Real-Time Data Integration for Data Warehouses, LNCS, vol. 2454, 2002, pp. 317–326, http://www.ifs.tuwien.ac.at/~bruckner/pubs/dawak2002_data_integration.pdf.

[38] M.A. Naeem, G. Dobbie, 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.

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

[40] S. YiChuan, X. Yao, Research of real-time data warehouse storage strategy based on multi-level caches, Phys. Proc. 25 (2012) 2315–2321, https://doi.org/10.1016/j.phpro.2012.03.390.

[41] I. MadeSukarsa, N. Wayan Wisswani, I.K.Gd. Darma Putra, L. Linawati, Change data capture on OLTP staging area for nearly real time data warehouse base on database trigger, Int. J. Comput. Appl. 52 (11) (2012) 32–37, https://doi.org/10.5120/8248-1762.

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

[43] R. Jia, S. Xu, C. Peng, Research on real time data warehouse architecture, Commun. Comput. Inf. Sci. 392 (2013) 333–342, https://doi.org/10.1007/978-3-642-53703-5_35, PART I.

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

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

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

[47] T.C. Chieu, 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.

[48] L. Chen, W. Rahayu, D. Taniar, Towards near real-time data warehousing, in: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE, 2010, pp. 1150–1157, http://ieeexplore.ieee.org/document/5474842/.

[49] C.R. Valencio, M.H. Marioto, G.F.D. Zafalon, J.M. Machado, J.C. Momente, Real time delta extraction based on triggers to support data warehousing, in: Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings, 2014, pp. 293–297.

[50] G.V. Machado, Í. Cunha, A.C. Pereira, L.B. Oliveira, DOD-ETL: distributed on-demand ETL for near real-time business intelligence, J. Internet Serv. Appl. 10 (1) (2019) 1–15, https://doi.org/10.1186/s13174-019-0121-z, arXiv:1907.06723.

[51] T. Freudenreich, P. Furtado, C. Koncilia, M. Thiele, F. Waas, R. Wrembel, An on-demand ELT architecture for real-time BI, Lect. Notes Bus. Inf. Proc. 154 (2013) 50–59, https://doi.org/10.1007/978-3-642-39872-8_4.

[52] M.M. N, D.R. K, A Novel Approach to Handle Huge Data for Refreshment Anomalies in Near Real-Time ETL Applications, vol. 1154, 2020, p. 545.

[53] S. Thulasiram, N. Ramaiah, Real Time Data Warehouse Updates Through Extraction-Transformation-Loading Process Using Change Data Capture Method. vol. 44, 2020.

[54] M. Toshev, Learning RabbitMQ, Packt Publishing, Birmingham, UK, 2015.