Physical Integration of Heterogeneous Web Based Data

Octavian DOSPINESCU, Sergiu CHIUCHIU
Alexandru Ioan Cuza University, Iasi, Romania
doctav@uaic.ro, sergiu.chiuchiu@gmail.com

With the ever-growing quantity of data available on the Internet and the development of new techniques to gain novel insights on the web of data, more attention must be given to the integration methods used. While the data extraction techniques have developed considerably with the rise of new automation tools that can be used even by persons without a software engineering background, the problem arises on integrating this data required for the analysis. In order to be able to gain valuable knowledge not only from a local perspective, but also from a global, enterprise-wide point of view and even from outside the borders of the organization, this data must first be integrated to a common storage capability of the business. The problem arises from the way most of the database management systems are designed, which in the early days of their development, it wasn't anticipated such an explosion of information from so many heterogeneous data sources. Moreover, there is an ongoing race on tapping into new information in places unexplored before, thus gaining valuable competitive advantage over other businesses. The problem of data integration is even more challenging in the case of web-based data sources due to the high frequency of changes that occur, meaning that solutions which works today are not guaranteed to work properly tomorrow.
Keywords: *Physical Data Integration, ETL systems, Data Collection Automation, Web of Data*

1 Introduction

Nowadays it is generally accepted that the main source of information is on the Internet. The problem is that while the quantity of data is ever increasing, the means of exploring this data has not developed at the same pace. According to [1], the percentage of data analyzed worldwide is less than 1% out of the total amount of data existent. All this while the quantity of data is increasing multiple times every few years according to [2], [3] and [4].

Since databases are not designed to work with heterogeneous data, as [5] mentions, an extra step is needed commonly known as data integration layer, in order to prepare the information for the analysis. This step can be implemented traditionally either by building a virtual mapping of the data sources that would offer the data on demand as a view [6], or through some means of physically integrating the heterogeneous data sources as described by [7] which would then be used to supply the decision support system with the required content. The advantage of the former method is that there is no need to store the actual integrated data separately, offering real-time perspective on the gathered information.

Even if data virtualization offers considerable benefits, the problem is that it cannot be used in every situation. This is the case for data gathered from the Internet, using different ways of collecting the information usually through some custom made web crawlers or by using automation tools that provides pre-defined procedures which help in speeding up the development process, while also reducing the number of error occurrences.

The reason why data virtualization is not a feasible solution for this case is because of several web specific factors. First of all, the data to be extracted is not stored in any database, rather most of the times we are dealing with semi-structured or even unstructured data which must be gathered from multiple web pages requiring considerable amount of time. This can also include images, videos or other large size documents that will impact data collection speed. Another factor refers to the problem of identifying and removing the duplicate records that might be present in multiple data sources, which will also impact the data retrieval speed thus making it unfeasible for virtualization.

DOI: 10.12948/issn14531305/23.4.2019.02

2 Data Integration Techniques

As stated by [8] there are multiple data integration techniques that can be used to obtain a centralized view on the required information. The most common integration techniques are: manual integration, middleware solutions, data virtualization and physical data integration/data warehousing.

2.1 The manual approach involves the user to collect the data and apply the validation and cleansing standards of the organization, followed by its loading into the database. It is only recommended for small datasets, or for handling exceptional cases which the integration software fails to treat. The drawbacks are the slower speed and the high costs per record, making it an unfeasible solution for large datasets.

2.2 Middleware solutions acts like a bridge between multiple data sources which allows both ways communication with the involved systems. By facilitating this communication it also implements data conversion, mapping and cleaning techniques. Middleware is highly recommended in case of large enterprises with multiple data source systems, where a central view is necessary for the monitoring of different business wide indicators which provides knowledge for the higher management. A well implemented middleware system can provide real-time information on the organization status and often represent a valuable comparative advantage over competition.

In contrast, it comes at a high cost and because of its inherent complexity it requires a lot of time to be implemented [9] resulting in a high chance that the project will end up in a failure.

2.3 Data virtualization represents an alternative to ETL systems that is continuing to grow in popularity, which unlike data warehouses it can provide real time information about the underlying systems. Furthermore, another big advantage of this technique is that data does not need to be copied in one single place like it is the case for physical data integration. This means that the required data remains in place and it is only accessed on demand. The scalability of a virtualization system which fairly easy allows the integration of numerous data sources is also an important benefit.

However, the main drawback of this method is that it consumes the resources that would otherwise be used to process operational changes, potentially negatively impacting them. For complex analysis that spans over long periods of time there is still the need for a data warehouse that offers superior performance for this type of operations as also emphasized by [10]. Moreover, another disadvantage is that not all software providers offer support for virtualization. Even if this technology in not a new one, this usually is the case for legacy systems found in large enterprise projects.

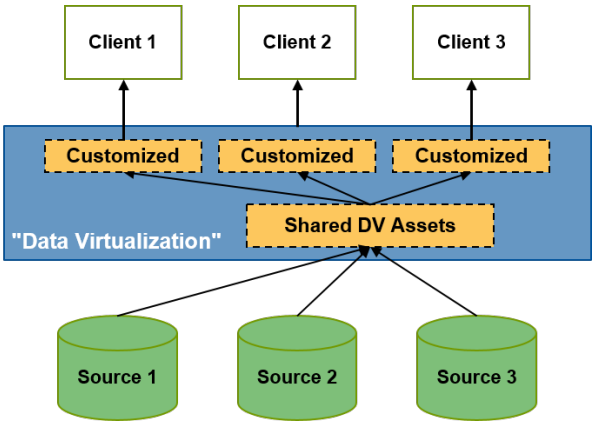

Fig. 1. Data virtualization architecture [11]

2.4 Physical data integration's main goal is to ensure a system that aggregates the information from all the other data sources inside or outside the organization in order to obtain a unified view, known as the "single version of truth". Data is not just retrieved as a view from the other systems, but it is copied into a separate database after a series of processing steps such as, cleaning, mapping, refactoring of data as also noted by [12]. Even if it is not designed to provide real time information over the integrated system, the advantage is that once the ETL process is completed, the user can rapidly and easily perform analyses and obtain reports for the decision makers even for large amounts of data. Another less noticed benefit is the possibility to version the data, which can be used for advanced analysis patterns or keep

DOI: 10.12948/issn14531305/23.4.2019.02

track of data evolution across time. There are 2 main physical data integration techniques: ETL (Extract, transform, load) and ELT (extract, load, transform).

ETL systems, are the most commonly used solutions of the two. Its main feature is the use of a separate intermediary environment before persisting everything into the database. According to [13], this system is responsible for the ETL processing and preparing the content to be persisted into a data warehouses. This supplementary step brings overhead time in the duration of the whole integration process since it is necessary to load data twice. Also, in case of complex transformations on a large amount of records the time needed increases as well.

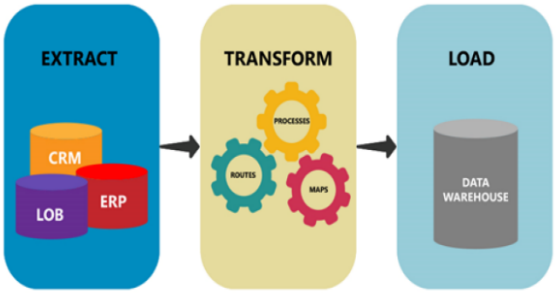

Fig. 2. Extract, Transform, Load process [14]

On the other hand, as also noticed by [15], the ELT system is designed specially to solve the problems raised from using an intermediary ETL environment, by directly loading everything into the data warehouse after the extraction phase. Then the data can be transformed straight from the data warehouse without the need of an intermediary system. Apart from speed, ELT additionally brings the advantage of flexibility by allowing the data to be persisted in the database in a rather unstructured format. This enables fast and easy use of data mining techniques that does not require structured data to work with.

However, this technology is still evolving, meaning that the options for this kind of tools is limited and comes at high costs. Furthermore, since it uses the data warehouse resources for the transformation stage, it means that for large sets of records it will impact other users using the same machine, possibly slowing down other processes. In general, ELT is a better approach when speed is of outmost importance and when the data does not need to be immediately transformed into a structured schema, or the complexity level of the computation needed is reduced.

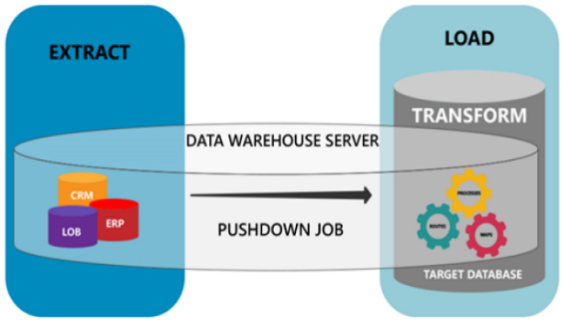

Fig. 3. Extract, Load and Transform process [14]

Physical data integration is generally the recommended option when there is a need for complex and time consuming analyses that requires data from heterogeneous sources, such that the operational systems are not impacted by conducting the analysis.

3 Physical Data Integration Application Model for Heterogeneous Web Data

Since the Internet has become one of the biggest sources of information nowadays, companies started to turn their attention, not only to the knowledge that can be derived from inside the organization, but also from outside of it. Taking this into consideration, the application model aims to illustrate how unstructured or semi-structured data from the internet can be extracted, processed and then loaded into a relational database management system. All of this without the need of human intervention in any of the stages.

The first part in the process involves collecting the data from the desired sources. This can be achieved either by developing from scratch of a custom solution or by using a specialized automation tool. In this case UiPath Studio automation tool was used, which has Visual Basic .NET as the underlying programming language. The reason for this choice is because it provides pre-built activities that allow us to easily collect the desired data from a web page, be it

DOI: 10.12948/issn14531305/23.4.2019.02

visible on the web page or not. Another reason for choosing this tool is the ability to create a software robot that can keep supplying the same results even in a dynamic environment that is continuously changing.

The data collection robot has been designed to gather specific information on various classified ads pages. As shown in figure 4, what the application does is to first go through the search results list of advertisements in order to get the URL of each page and then save them in a CSV file. Using the URLs provided in this file each page will then be accessed sequentially.

The second part of the process opens a new browser instance, builds an in-memory data table and then collects the required data from each page in order to populate the data table.

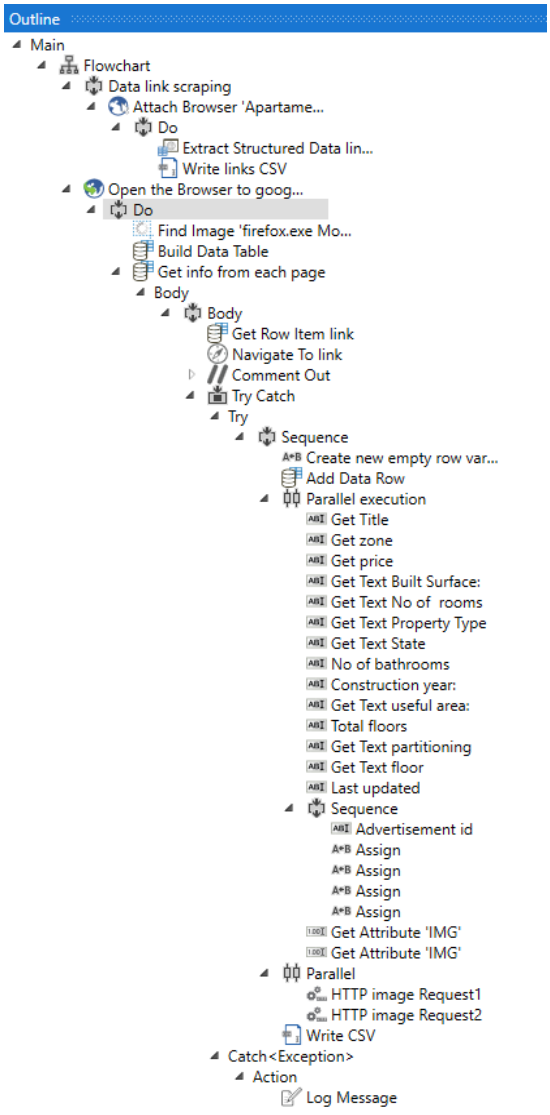

Fig. 4. Web crawler logical structure

For downloading the images, a different approach had to be taken. First the URLs of the images are gathered from the page and then various GET request parameters are modified in order to receive the images from the server at their original size. After the URL has been properly formatted, a HTTP GET request can be initialized in order to receive the image. In the database, only the image file name will be persisted. After all the data is extracted from the page the record is saved into a CSV file.

A commonly occurring problem when designing such a program workflow is forgetting to make the application, exception recoverable. Because it is common to expect that in some cases, part of the data that needs to be collected from the page, is missing. In order to prevent such exceptions from terminating the entire application, these type of activities must be enclosed in a try-catch block and to prevent slowing down the entire process the default timeout should be reduced to 3 seconds. Moreover, the amount of data processing in this step must be kept to a minimum in order to maximize the performance.

After the desired data has been successfully collected from the web, the ETL process can begin. For this part, a custom Java application has been developed. The database chosen for the persistence layer is PostgreSQL 10, since it is a lighter solution in comparison to other alternatives such as Oracle database, yet it offers competitive performance and similar functionalities.

To ensure a faster and less error prone application development a few additional dependencies have been added to the project.

- *Spring Boot* framework, in order to use the dependency injection mechanism provided
- *Spring Repository*, in order to use the already implemented repository pattern for database manipulation
- *Project Lombok* which generates the Java beans getters, setters and constructors at compile time, without being necessary to explicitly declare them in the classes. It was also used to

DOI: 10.12948/issn14531305/23.4.2019.02

implement the Builder pattern such that new instances can easily be created with the desired values.

- *Open CSV* used to iterate through the CSV records and extract the data from this type of files.
- *Model mapper* that is useful when mapping already existing records from the database with the updated information

3.1 Extraction phase:

The first feature that must be implemented in the application is the data extraction process. Since the data is provided in a format that is following the CSV specifications, we will use Open CSV library which is already added as a Maven dependency into the project. This library helps us to create a CSV reader that is used to iterate through the lines of the file being read. First of all an input stream is opened towards the file to be read using an instance of BufferedReader class, which is then passed to the integration service. Since the application is designed to deal with multiple data sources, depending on which type of data source the application is initialized with, it will choose the right integration service at runtime as it can also be seen in figure 5.

```
public void execute() {
    BufferedReader br;
    File dataFileCSV;
    try {
        switch (dataSource) {
            case Constants.DATA_SOURCE_M:
                dataFileCSV = new File("myData.csv");
                br = new BufferedReader(new FileReader(dataFileCSV));
                integrationServiceM.mapStreamToEntities(br);
                break;
            case Constants.DATA_SOURCE_T:
                dataFileCSV = new File("myDataSt.csv");
                br = new BufferedReader(new FileReader(dataFileCSV));
                integrationServiceT.mapStreamToEntities(br);
                break;
            default:
                LOG.error("Unknown data source type. Cannot continue the " +
                        "integration. Process aborted...");
                break;
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
```

Fig. 5. Opening a file input stream and passing it to the integration service

3.2 Transformation and Loading Phase

The integration service structure was designed taking into consideration code reusability, avoiding duplicate code as much as possible. Because of this, the service methods are exposed via an interface, which is implemented by an abstract class that holds the methods that offer common behavior. Further on, the abstract class is extended by concrete classes that holds the specific implementation for each data source type, as it can be seen in Figure 6.

DOI: 10.12948/issn14531305/23.4.2019.02

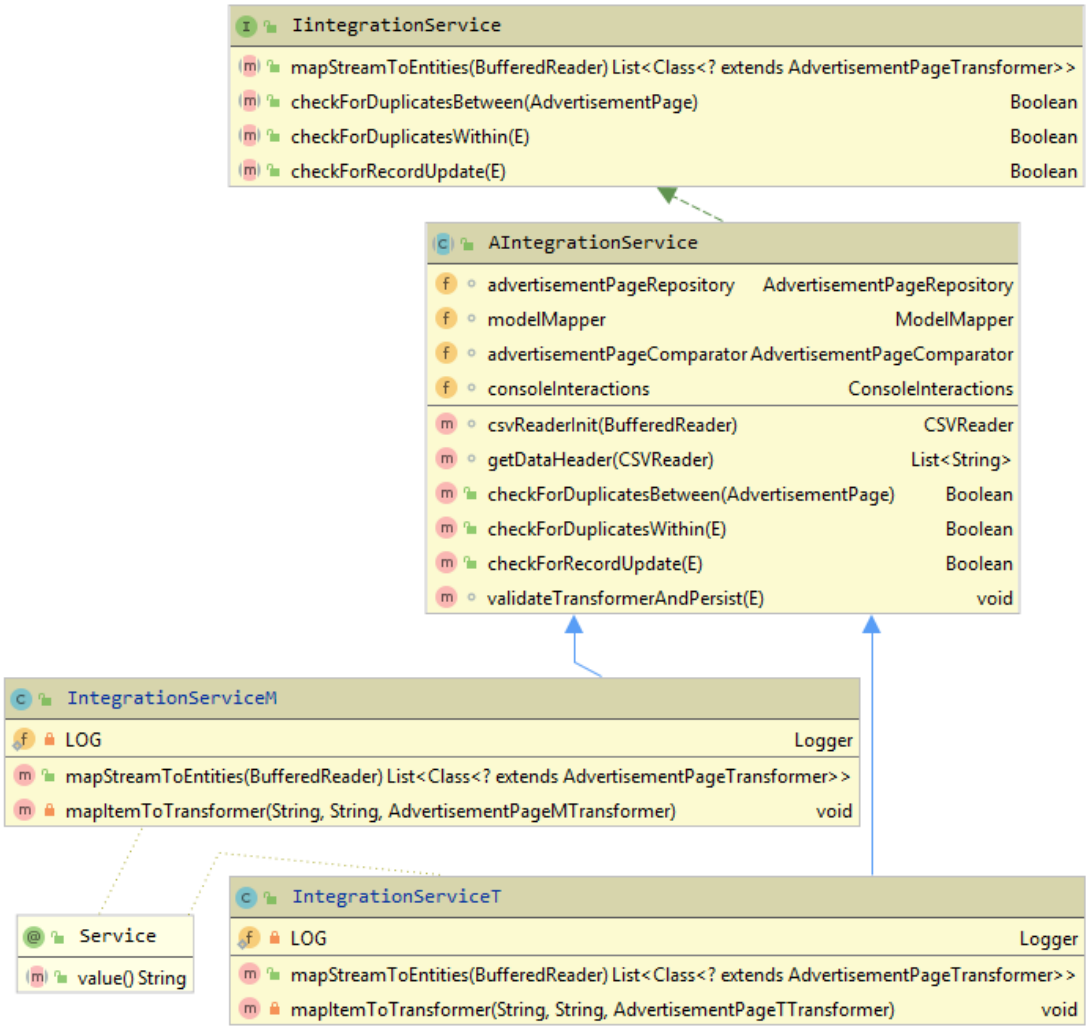

Fig. 6. Integration service hierarchy diagram

The first method from the integration service that must be called is mapStreamToEntities. This method's role is to iterate through each CSV line and for each of the lines it will further pass the responsibility to transformation and validation methods. By doing the processing line by line rather than in batches, it allows to easily scale the integration effort in the future by running parallel threads that reads and process multiple data rows at the same time.

After the CSV line is read from the input stream, the raw data fields are mapped to an intermediary transformer entity using mapItemToTransformer method. During the mapping process, the data is also transformed, cleared of unnecessary content, converted from String to the appropriate data type and mapped to the corresponding entity field. When this process is finished, a populated AdvertisementPageTransformer instance is obtained depending on the data source used, whose fields are then compared with the records already persisted in the database. The first validation that it passes through checks whether or not the new record pageId already exists in the database. If it is already existent, then it is verified if the record represents a more recent update than the one already existent in the database. When this is the case, then the transformer instance is converted into a new instance of AdvertisementPage bean. Next, the record to be updated is loaded from the database into the application, then using Model Mapper the fields are automatically updated.

DOI: 10.12948/issn14531305/23.4.2019.02

```
<E extends AdvertisementPageTransformer> void validateTransformerAndPersist(E
transformer) {
    Boolean isDuplicateWithin = checkForDuplicatesWithin(transformer);
    if (isDuplicateWithin) {
        Boolean isModifiedRecord = checkForRecordUpdate(transformer);
        if (isModifiedRecord) {
            AdvertisementPage updatedAp = transformer.mapTransformerToEntity();
            AdvertisementPage ap =
advertisementPageRepository.findByPageId(updatedAp.getPageId());
            modelMapper.map(updatedAp, ap);
            advertisementPageRepository.save(ap);
        }
    } else {
        AdvertisementPage ap = transformer.mapTransformerToEntity();
        Boolean shouldSave = checkForDuplicatesBetween(ap);
        if (shouldSave) {
            advertisementPageRepository.save(ap);
        }
    }
}
```

Fig. 7. Validation method

When there is no similar pageId found in the database then it will proceed with a more fine-grained duplicate validation. This will compare the most relevant fields of the newly obtained entity with the corresponding values of the records already existent in the database. In order to achieve this functionality a comparator class has been developed. This class contains special methods for each type of field in order to check for equality and save the answer in a Map variable. After the comparison is performed and the answers are saved in the HashMap instance, a duplication percentage score is calculated based on how relevant is each of the fields found to be equal. An attribute that it is known to naturally hold only a few elements in its possible values range, will have a smaller impact on the duplicate score, in comparison to an attribute that usually holds distinct values.

The way the duplicate percentage score works is by creating a weighted sum according to each field relevance, from all the compared attributes that are found to be equal. This sum is then divided to the total potential score, that is, the sum obtained if every field compared would turn out to be equal.

After the percentage score is obtained, it is compared to a pre-established threshold. If the percentage value is below that threshold then the new record is not considered a duplicate and is persisted into the database. Otherwise, the new record can automatically be discarded without being persisted into the database.

Another option for this case that was also implemented in the application is to prompt the user on the console that one of the new records is similar with another one already existent into the database. It then shows the duplication percentage, instance variable values for both objects and prompts the user to decide whether or not the new data should be saved. After these actions are performed the application moves on to a new CSV line and repeats the process until all input is finished.

DOI: 10.12948/issn14531305/23.4.2019.02

```
private Double calculateDuplicateScore(Map<String, Boolean> duplicateFields) {
    Double score = 0.0;
    Double maxScore = 0.0;
    Map<String[], Double> categoryScore = new HashMap<>();
    categoryScore.put(highScore, FIFTEEN);
    categoryScore.put(mediumScore, TEN);
    categoryScore.put(lowScore, FIVE_AND_HALF);
    for (String[] category : categoryScore.keySet()) {
        for (String field : category) {
            if (duplicateFields.get(field)) {
                score += categoryScore.get(category);
            }
        }
        maxScore += categoryScore.get(category) * category.length;
    }
    return score / maxScore;
}
```

Fig. 8. CalculateDuplicateScore method

4 Conclusions and future directions

As it was emphasized throughout the paper, the importance of unstructured and semi-structured data collection from the internet is becoming increasingly important for businesses to keep their competitive advantage. Due to this, the application represents a realistic sample of what could be turned into a large scale custom developed integration tool that would meet all the necessities of a business.

The application contains a series of benefits since it has been designed to easily plug in the integration logic for additional data sources. Furthermore, by opting for a line by line reading and processing rather than executing these operations in batch, it allows the application to be easily scaled up for processing larger amounts of data by using multiple threads. In addition, it represents a viable solution that offers speed and flexibility in development by implementing appropriate architectural designs using the latest technologies and tools available.

In the future versions of the application a format agnostic data input reader can be implemented, which will be able to accept as input at runtime multiple file types. Moreover, a user interface could be developed that would offer the user the possibility to customize different aspects such as having the possibility to decide which fields to persist in case of duplicate records or perform batch operations on the duplicates found. Another useful feature that could be included in the project is the ability to compare and find near duplicate images using machine learning algorithms.

Acknowledgments: The authors intend to include the results of this research paper in a master thesis that will be presented at Software Development and Business Information Systems master's final exam in July, 2020.

References

[1] J. Gantz, D. Reinsel, *"The Digital Universe In 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East"*, IDC's Digital Universe Study, IDC 1414_v3, pp 2-3, December 2012

[2] T. Barnett, Jr. et al., "Cisco Visual Networking Index: Complete Forecast Update, 2017–2022", pp. 67-79, December 2018

[3] D. Reinsel, J. Gantz, J. Rydning, "The Digitization of the World From Edge to Core", IDC White Paper – Doc# US44413318, pp. 6-11, November 2018

[4] H. Guo, Lizhe Wang, Fang Chen, Dong Liang, *"Scientific big data and Digital Earth"*, Chinese Journal, DOI 10.1007/s11434-014-0645-3, pp. 5066-

5069, 2014
[5] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis, *"Fundamentals of Data Warehouses"*, Springer-Verlag, 2000
[6] D. J. Ullman, *"Information Integration Using Logical Views"*, InProc. ICDT97, pages 18–36, 1997
[7] P. Ponniah, *"Data Warehousing Fundamentals for it Professionals, Second Edition"*, John Wiley & Sons, Inc., Hoboken, New Jersey, DOI:10.1002/9780470604137, 2010
[8] A. Reeve, *"Managing Data in Motion"*, Elsevier, pp. 15-17, 2013
[9] S. Krakowiak, *"Middleware Architecture with Patterns and Frameworks"*, pp. 8-35, 2009
[10] L.J. Pullokkaran, *"Analysis of Data Virtualization & Enterprise Data Standardization in Business Intelligence"*, Massachusetts Institute of Technology, pp. 55-58, May 2013
[11] M. Selvage, *"Design a Resilient Data Virtualization Architecture"*, July 2016
[12] P. Ziegler, K. R. Dittrich, *„ Data Integration — Problems, Approaches, and Perspectives "*, University of Zurich, pp. 5-7
[13] Q. Liu, *"A Design of ETL for the Construction of Traffic Network Based on Big Data"*, CET VOL. 51, pp. 451-453, 2016
[14] I. A. Alvi, *"ETL vs. ELT: Transform First or Transform Later?"*, 2018, Available: https://datawarehouseinfo.com/etl-vs-elt-transform-first-or-transform-later
[15] R. J. Davenport, *"ETL vs ELT, A Subjective View"*, Commercial Aspects of BI discussion papers, pp. 9-10, June 2008

Octavian DOSPINESCU graduated the Faculty of Economics and Business Administration in 2000 and the Faculty of Informatics in 2001. He achieved the PhD in 2009 and he has published as author or co-author over 30 articles. He is author and co-author of 10 books and teaches as an associate professor in the Department of Information Systems of the Faculty of Economics and Business Administration, University Alexandru Ioan Cuza, Iasi. Since 2010 he has been a Microsoft Certified Professional, Dynamics Navision, Trade&Inventory Module. In 2014 he successfully completed the course "Programming Mobile Applications for Android Handheld Systems" authorized by Maryland University. He is interested in mobile devices software, computer programming and decision support systems.

Sergiu-Georgian CHIUCHIU is a student attending the Master of Software Development and Business Information Systems at the Faculty of Economics and Business Administration, within Alexandru Ioan Cuza University of Iasi. During the years 2016-2017 he also participated in an Erasmus exchange project where he studied for one year at University of Freiburg. He is also working as a software developer in various projects for multinational companies. His interests include web applications development, machine learning and robotic process automation.

DOI: 10.12948/issn14531305/23.4.2019.02