ETL as a Necessity for Business Architectures

Aurelian TITIRISCA
University of Economic Studies, Bucharest, Romania
aureliantitirisca@yahoo.com

Today, the new trend that companies are following is to digitalize all data they have in order to reduce costs with physical space and better handle data volume. The new era, the era of bundling data sources and huge volumes of data continues this decade also. In all industries, companies are starting to understand and appreciate the value of data, the value that these volumes of data can produce.
Companies are collecting huge volumes of data but not always the actual solution for business intelligence (BI) can handle these volumes. To obtain information, those volumes of data are analyzed with extract-transform-load (ETL) software solutions. Companies, in the current economic context, are finding hard to invest in improvements of BI process including in ETL process.
In this paper I will demonstrate why this kind of investment is necessary and also I will demonstrate that ETL process must be included in BI and big data architectures. In the following pages I will refer to business architectures as BI and big data architectures.

Keywords: *ETL, business intelligence, Big Data, Internet, dataset*

1 Introduction

Today, companies are using World Wide Web to promote themselves, to find huge data sources, to educate their staff and they are using it as a market (e-commerce). Internet has revolutionized the *information domain*: obtaining data from anywhere at lower cost, finding new data types, transporting the information where needed, analyze the input data and making decisions that in the end will offer more data to be analyzed. And we must not forget that Internet made possible online commerce and online transaction. The entire business intelligence infrastructure is based on the World Wide Web and most important BI is all about data and analyze data.
The data volume that companies are collecting is growing exponentially. The firms can have access to a bundling number of data sources inside and outside their business. They need to capture the data (that in majority is form by not structured data), to organize it, to store it, to transform and analyze data, to distribute the obtained information by users and their roles. This is the competitive advantage but also the challenge that companies must answer.
In the current economic context, the survival of a company depends on the rapidity in recognizing a change in the dynamic business environment and the reaction to create a correct response to those changes. Every company to succeed in achieving goals must anticipate the market evolution, find and explore new trends in the market, reorganize the BI strategy and reallocate resources to have a competitive advantage over its competitors. All of this have as a basic layer the information and the value that can be obtain by it. BI architecture and big data architecture must include extract-transform-load (ETL) solution for manipulating volumes of data between source and data warehouse, between data warehouse and data marts.

2. Business intelligence and big data with ETL solution

The trend that companies followed in the last years is to digitalize all data that they have, entire document flows. This way they can reduce the cost with storing data but also increase the data volume from which the current BI solution is trying to retrieve valuable information. For example, in electricity domain, the companies from United States of America (SUA) estimate that in 2019 will install over 53 millions of devices for measuring the consumption. Those devices will send information in a time interval (once a week, twice a week, once a month, every six months or one year) and in the end the volume of data that needs to be processed will increase.

I want to make a parenthesis to remind why BI is so important, and refer to reference [1]. BI helps economic agents to create synthesized information out of data sources, to use the obtained information in the decisional process and to accomplish business goals. In addition, BI helps economic agents to:

1) identify errors in the companies - accounting errors, errors in selling department, errors in the document flow;
2) discover new opportunities of growing - by studding the market and find those market sectors that the company could explore but it does not. For example, in the retail, we find that there is a demand for smart watches that the competition does not cover;
3) understanding competition: identify the competition, market position;
4) identify which departments are performing below average and why;
5) identify which departments are exceeding the performance and why;
6) identify consumer behavior - by analyzing sales records and by market polls;
7) define performance indicators - like employee score card, productivity, claims;
8) adapt BI process to the market needs - for example adapt the stocks to demand;
9) better understand the economic period and the market - in this case, in retail, the company can rise or lower prices depending on the customer revenue;
10) analyze historical data generated by the business process or from other data sources like national institutes of weather (climate changes), statistical data - by analyzing this data a retail company can find opportunities in the market, can over supply a store before floods. Another example, in the green energy segment, companies can identify the zones in which they can produce more energy from natural resources.

BI solutions continue to evolve. When a company wishes to adopt a BI solution, I believe that the first step is to identify the data flow and the data sources that the company can access, identify possible data sources that can bring value to the company.

Mike Ferguson in his paper work "Architecting A Big Data Platform for Analytics" [2], considers that big data contains huge volume of data and the IT solutions for handling those volumes of data. And I agree. It is the most basic principle, because basically we can only handle big data with the correct IT solution in order to obtain correct information fast.

The IT companies that offer business consulting divide their solution in order to cover a bigger part of the volume of data offered by clients. Today, not all companies offer solutions for big data. Big data represents a huge volume of data, from different sources (like multimedia data, social data, emails, phone records, commands on email), a data that in majority is not structured.

In my opinion, big data can be represented by volumes of data that companies cannot handle and from which it cannot obtain any information or obtaining information from

big data takes too long. From my practical experience, I can say that obtaining information in over 10 minutes from a relative large volume like terabyte is already too much and requires a new BI solution.

From volumes of data we can obtain valuable information known also as business value added (BVA). BVA helps companies to reach their target because the volume of data grows every year in all domains and it is hard to analyze them with the current IT solution that the company has.

From my point of view all architectural schemas of BI have to use a ETL solution to manipulate and aggregate data. A basic architectural schema can be describe in figure 1:

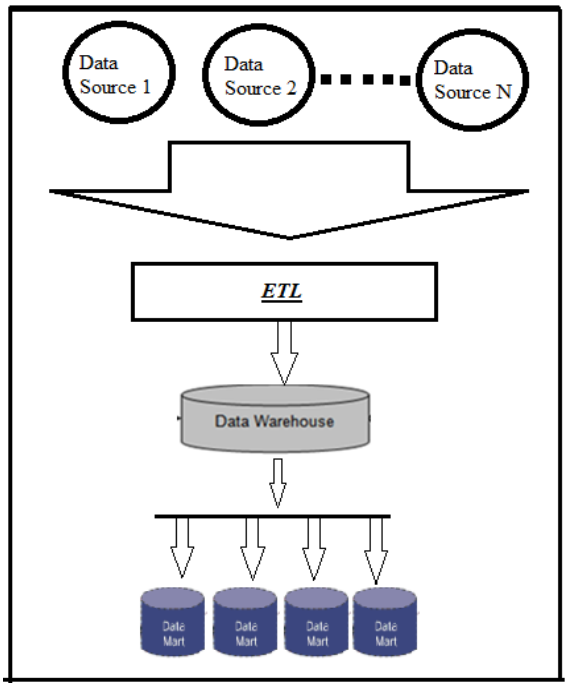

Fig 1. ETL in BI and big data architecture, from my point of view.

Usually an architecture has one or more data warehouse dedicated for collecting data (first level), another data warehouse or one or more data marts for storing the data after the ETL process (second level) and at a third level, data marts dedicated for storing the obtained information. Between the second and third level we can have another ETL layer. ETL processes are used for the alimentation of data warehouse.

Nowadays, BI is not a optional decision that companies could take but is a mandatory one that companies must choose in order to survive on the market, it is the way that they can obtain advantages and evolve on the market.

From my point of view, managers that have access to accurate information and correct synthesized information have a bigger chance to create better decisions, to anticipate the market. Regardless of technological evolution, the most important value that BI can offer is information and the correct usage of the information is the most important competitive advantage.

Today, on the market there are present a number of big data solutions, from which I will highlight an open source solution (Cloudera) and an IBM solution (Big Insights). Big data appears when the volume of data exceeds terabytes of data (petabytes, zetabytes, exabytes). Big data is the new challenge that IT domain are trying to solve.

3. Extract-Transform-Load data

Unfortunately, the data quality from data sources is not at its best and definitely does not map on a metadata schema of a data warehouse. This is way we need ETL process for cleaning the input data that the business can access. ETL means extract-transform-load data. There are three steps in manipulating data.

The first step is to extract all data from data sources. Because the extracted volume of data is unpredictable, it would be contra-productive and time-consuming to extract this volume each time that a superior step fails. This is why the volume of data should be stored into a binary file, internal file recognize by the ETL product. When extracting data, it is wise to define a perimeter for the input data. For example, we will treat all data received in a period of

time. We can do this because a ETL product offers an operator called *delta*, that has the role for identifying new records, old records that have to be updated or records that need to be deleted. This operator works on keys: identifies if the key is in the target. If not, it creates it. If already exists, then it compares values for each attribute. If there are changes then we update the data in target. If one key (from the point of view of value) from target does not exist in key source then we delete the data from target. An important observation regarding the operator is that the input fluxes must be sorted and partitioned by key.
The basic job design for data extraction is described in figure 2 below.

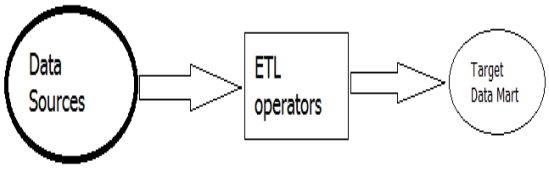

Fig 2. Basic job design

In the practical example, Infosphere Datastage uses personal binary files, called dataset. These files are kept on the Infosphere Information Server. Datasets are binary files that can hold metadata, keep sorting and partitioning of the data and data. And because it is a binary file, operations of writing and reading data from server are fast. Also this type of file can be used by any interface of a module, if the business requires it. For maintenance, modules are easy to be handled if the three steps are well identified at a interface level.
The second step is for transforming data. In this step, we can do cleaning operations on data. The simplest operations are: character conversion, type conversion, normalizing data so we could map the data to the target metadata, define keys, treat null values, conversions, access analytics function. At this level we can aggregate the data to obtain the desired information. At this step we can use the *delta* operator or the *slowly changing dimensions which is similar to delta.*
It is important that the ETL maintains the integrity of the references so we can eliminate the risk of referring a table, a dimension that doesn't exist. ETL process must assure that the correct information is obtained and loaded into the data warehouse, from business logic point of view.
From the point of view of Infosphere Datastage, the fastest way to load data into tables is with the option load truncate, because it executes two fast SQL operations: truncate data from the table and insert the new data into the table.
From my point of view, before we can load data into the target table, target data mart or data warehouse it is important to verify that the new data fulfills the following conditions:
1) referential conditions: we must verify if the new data respects the architectural schema of the data warehouse;
2) join conditions: it is important that the joins respects the architectural schema and the business conditions;
3) unique conditions for attributes: identify witch attributes are keys or have unique values;
4) null conditions: verify witch attributes can receive null as a value and for which we treat nulls;
5) dropping and rebuild indexes, triggers on tables;
6) respect the specifications for developing and the mapping.

The third step is dedicated to load the dates from step 2 into tables, data marts, data warehouse. When we load data into tables, the user for ETL must have roles that allow him full access to the target table. In this step it is necessary to treat the fail module condition: if a module for actualization of

data fails then it is important to delete the inserted data and reload the data.
Although an ETL solution seems easy to be used, in practice is the other way around. Over 50% of the time is allocate to map the ETL process to business needs.

4. Code reuse in ETL products. Import-export operations in ETL

In the last years, using of ETL has had a code reuse problem. When we refer at ETL, the term of code reuse cannot be used because each job that developers create is unique and treats in a unique way a business process. Every interface has its own specifications although this must not be confused with similarities between interfaces.
From my point of view, when I create an interface I have in mind the specification, the unique metadata and unique names of the tables from witch I extract and load data. In addition, in practice the test for an interface are done individually because if, by absurd the interface is not unique in same way, it's like a single interface can treat all BI process and it will mean that the data warehouse contains one dimension.
For example, in a retail, one business requirement says that a dimension of the data warehouses must not contain null values and must have all rows identify by a unique number. In order to solve the business need, the ETL developers need to develop the following modules:
1) a module for extracting the data: data is extracted from the data warehouse with a specific purpose;
2) a module for treating all nulls cases, for identifying actualization operations;
3) a module for each type of actualization of the target table.

From my point of view this particularity on modules instead of one big module is very useful for maintenance operations.

Observations in some ETL products, modules are referred as jobs. Also operators and operations are included in stages and stages functionality. A collection of grouped jobs form an interface. Actualization operations on a data warehouse refers to identify new data that doesn't exist in the target table, identify all data that need to be updated, data that need to be deleted from the target table or data that is copied into the target table.
When comes to manipulate data throw ETL, there are two principles that parallel and sequential processing need: partitioning and collecting data. The collecting operation means that data from different nodes is collected into one output node. Collecting data in a specific order can improve performance and reduce sort operations.
The partitioning operation means that data is distributed on the processing nodes by a predefined algorithm. Most use algorithms are:
1) round robin: distributes the data evenly on the nodes;
2) hash: distribute data on nodes by key or group of keys;
3) same: keeps data partitioning from previous stage.

Developers have created interfaces and modules from scratch. I consider that there is a way to reuse the code if the business rules in essence are almost the same for two or three target dimensions. Here I refer to the fact that two interfaces can have the same principles: extract the data into a file in the first module, treat the data for null values and then identify the actualization (similar business rules) operations on the target table and the last module for applying those operations on the target table. We can reuse those modules by creating copies of the models of the previous interface, reuse the operators and operations, parameters but we cannot use the same metadata, the same name for the jobs and the same intermediate files.

Depending on the ETL product, we can have stages similar to global variables called global shared container. The shared container has the commune keys and fields of each interface and the same business rules. The rest of the columns are mapped for each interface. For example, Datastage uses shared container and for mapping the other columns that are not keys has a run time column propagation property (columns are past at run time).

From my point of view, I consider code reuse in ETL products to be a double blade sword. It can be indicated, because it reduces time of developing, it can lead to identify errors from grammatically errors (for example wrong file name, wrong name of a stage variable, of a parameter) to calculation errors (some formulas or cases are not treated; the mapping of columns is wrong, one integer field is mapped to wrong field; the reject file does not contain the correct information). Another hand, if the developer is not paying attention he could propagate the error to more interfaces and the human error risk can grow.
ETL programs can import and export code. Throw code I will include interfaces, modules, definitions files. Usually, an ETL program can export and import definitions files in internal format, for code protection and to prevent the client to choose another ETL solution. But there are ETL solutions, like Infosphere Datastage that can import and export files in internal format, called *dsx*, or xml files. A *dsx* file contains the interface or modules as precompiled code similar to C language and contains: metadata, operations grouped by stages, name of links, input and output files and number of nodes for process data.

5. Company motivation to update ETL solution

ETL solutions are important for all BI process. They are a part of BI architecture. Although the cost of a ETL solution is not small it is a mandatory part for each BI process. Companies have to take into consideration the cost of the investment, the depreciation rate and the benefits of the ETL product. Normally, you cannot use a ETL without the proper training and that includes training cost or support team cost.
ETL solutions evolves all the time from architectural point of view (execute modules from sequential to parallel), calculation power, scalability, real time support. IT companies sell license for old and new versions of the ETL tool. There is a challenge to convince clients to update the ETL version because this operation, in majority of cases, includes additional costs generated by incompatibility between versions and by clients needs in using new facilities. It needs a maintenance team to handle the migration process.
The migration process is not a short term process because the team needs to analyze the impact that the new solution will have and solve the eventual errors. During the migration process both ETL versions will run in parallel until the migration is complete Customers are not always willing to pay license for both version and support. This is way investing in a evolution of a ETL product needs to be analyze correctly. Another reason that migration process needs a support team is that of architectural change of the newer version that will cause imported modules with join operations between tables to generate a false result.
The trend in ETL solutions targets the need of developers to write code: the developer should not need to write code by himself, instead he should use the improvements of the new version.
From my point of view the investment cost in a ETL solution should be very well supported. Companies must analyze the

impact, the benefit that the new solution brings to the BI process. There are cases in which the BI process requires a new feature like real time processing data that the old version does not support.

Throw the research that I made, I will demonstrate why a investment in the evolution of ETL solution is required for the BI process. I analyzed two versions of the ETL - Infosphere Datastage: version 7.5.3 and version 8.5. To better understand this version and the capabilities, you can read references [3] and [4].

Infosphere Datastage is a ETL product own by IBM. Is a graphic program for analyzing and data manipulation. From the operating system point of view, the Infosphere Information Server runs on UNIX and the Infosphere Datastage client runs on Windows. At ETL level, exists the following jobs status: *finished, aborted, not compiled, finished (see log).*

Datastage offers the following advantages:

1) parallel processing capability;
2) metadata manipulation;
3) support for data in real time;
4) support for BI rules with any grade of difficulty;
5) allows direct access to big volumes of data;
6) offers public support: IBM manuals, big community support;
7) export and import data throw xml files;
8) offers connectors for each database (DB2, Oracle, Netezza, Teradata);
9) offers access to modules logs after the execution;
10) can produce executable files throw the export options that he has;
11) can execute SQL queries;
12) allows metadata import from different sources with the exceptions of data types BLOB, CLOB.

I will analyze the Oracle Connector stage from the points of: execution mode, data type conversion and null conversion. In my example, I defined an Oracle table, PERSOANE with attributes:

```
CREATE TABLE PERSOANE(
IDI_PERS NUMBER(11) NOT NULL,
COD_PERS NUMBER(11) NOT NULL,
DAT_DERN_MAJ_ACTIV_CLI DATE,
PERS_EMAIL CHAR(1 CHAR),
DAT_MAJ_EMAIL DATE,
PERS_EMAIL_PART CHAR(1 CHAR),
DAT_MAJ_EMAIL_PART DATE,
PERS_TEL_MOB CHAR(1 CHAR),
DAT_MAJ_TEL_MOB DATE,
DAT_HEUR_CREA_TECH DATE NOT NULL,
DAT_HEUR_DERN_MODIF_TECH DATE NOT NULL,
DAT_MAJ_TOP_CLI_MIGR DATE,
)
```

On the canvas, I added a connector stage and a dataset file, like in figure 3, bellow.

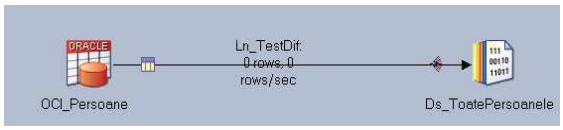

Fig 3. Job designed by me in Datastage 7.5.3.

In Datastage 8.5, the job design is:

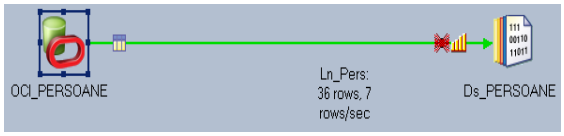

Fig 4. Job designed by me in Datastage 8.5

From execution point of view, the Oracle connector in Datastage 7.5.3 is set by default to run in sequential mode, shown by the figure 5 below in tag *execution mode*:

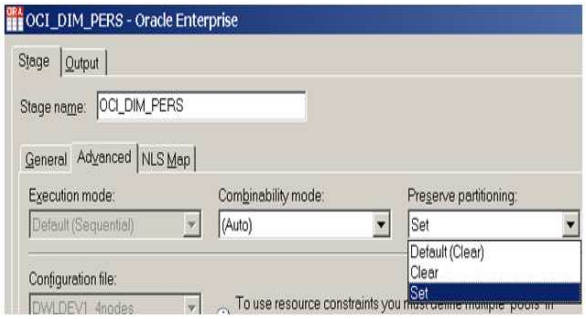

Fig 5. Oracle connector in 7.5.3 from execution point of view

In Datastage 8.5, the Oracle connector can is default set to run in parallel but without partitioned read method set to true, it runs in sequential as showed in the log and imagines below (figure 6). It can be forced to run in parallel by using the option: *Enable partitioned reads* set to yes and *Partitioning read method* set to yes.

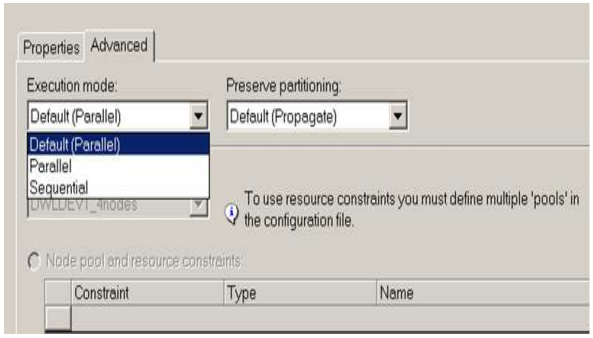

Fig 6. The Oracle connector in 8.5

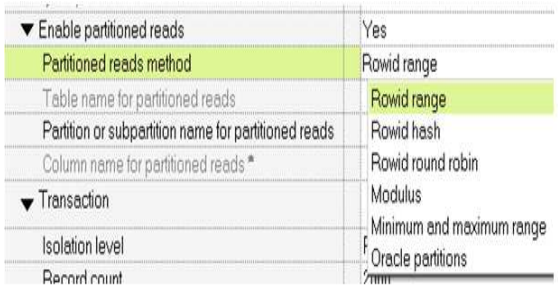

Fig 7. Configure read in parallel operation.

Without this configuration the Oracle connector, although it suppose to run in parallel, notify the user in job log that it will run in sequential mode:

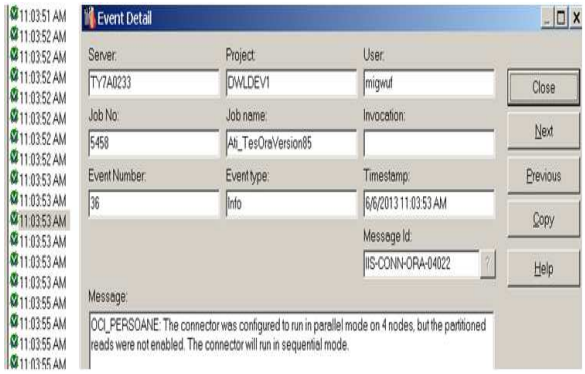

Fig 8. Oracle connector needs the partitioned read option to be enabled

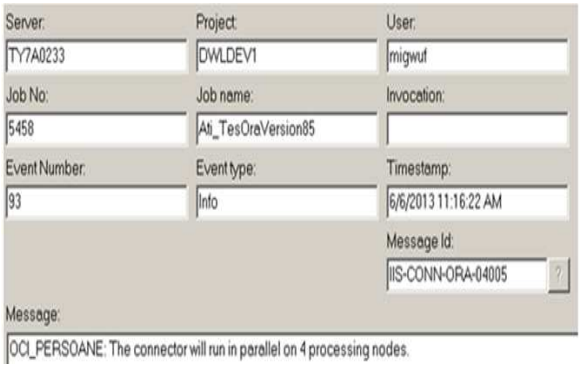

Fig 9. Oracle Connector extracts data in parallel.

Each table includes an attribute that contains the row number. When we use Rowid Range method, the Oracle connector follows a execution plan:

1) the connector queries the DBA_EXTENTS dictionary to obtain information on the table;
2) connector then defines a ROWID value for each processing node based on the results obtain above;
3) at execution, each node runs a query with a personalized where clause.

This method doesn't work if:

1) if the user cannot access the DBA_EXTENTS dictionary;
2) if the connector reads from an indexed table or view;
3) if the SQL script contains joins between tables.

I will describe the results obtained in Datastage 7.5.3 and in Datastage 8.5 when I change the null ability of attribute DAT_MAJ_TOP_CLI_MIGR to no. The job does not extracts any data because null values cannot be inserted into a not null attribute. And both version will act in the same way as showed in the imagines below.

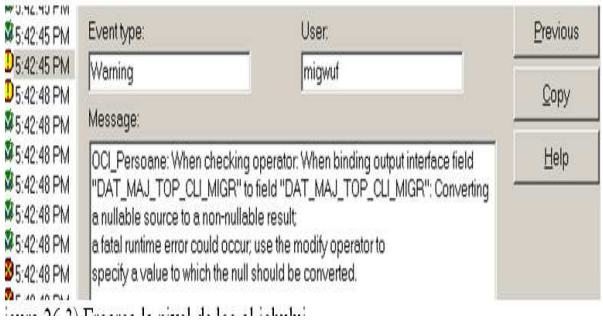

Fig 10. Log message in Datastage 7.5.3

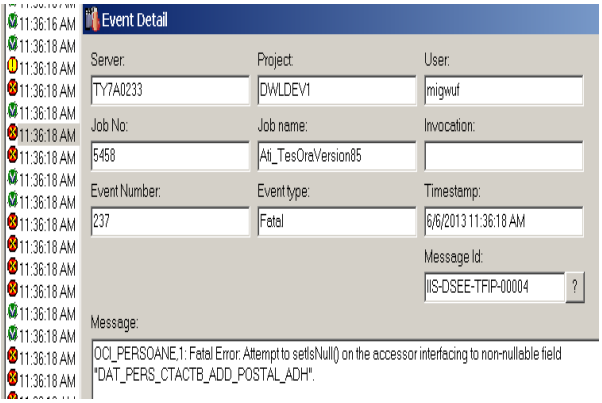

Fig 11. Log message in Datastage 8.5

I will show the result obtained when we are trying to map a timestamp attribute from Oracle to a string attribute in Datastage. In version 7.5.3 this conversion is not done automatically and the developer must do it manually in a Transformer stage or Modify stage. Datastage 8.5 treats this conversion from timestamp to string automatically as showed in the below log messages.

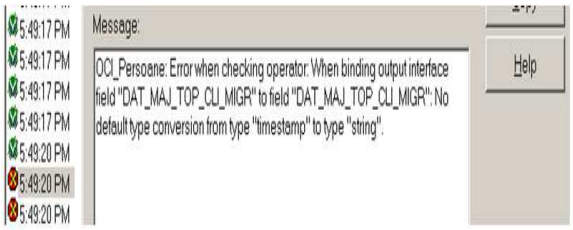

Fig 12. Log message in Datastage 7.5.3.

See the red *X* from above that indicates that the job failed.

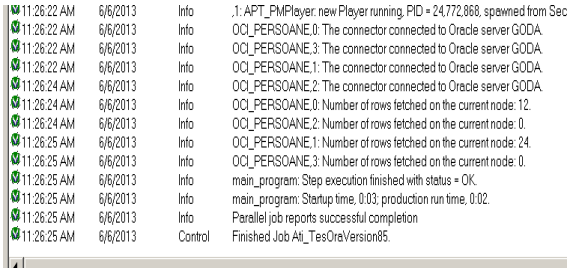

Fig 13. Log message in Datastage 8.5

I will show the result obtained when we are trying to map a timestamp attribute (in format DD/MM/YYY 00:00:00) from Oracle to a date attribute in Datastage.

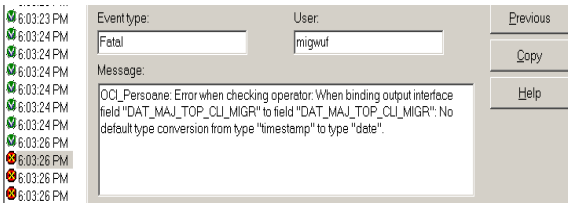

Fig 14. Message log in Datastage 7.5.3 in the above case.

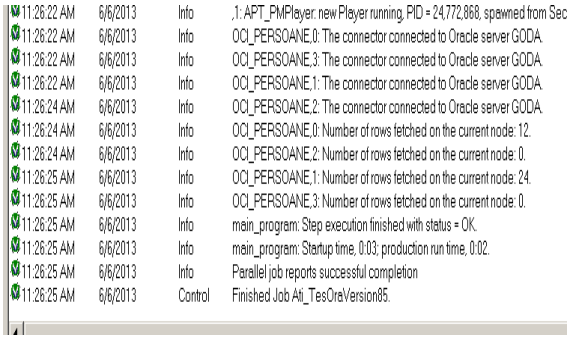

Fig 15. Message log in Datastage 8.5

In this case, the job was executed successful. I can say that my study shows the sensibility of the ETL tools at data types imported from Oracle.

6. Conclusions

The bellow table summarizes the obtained results:

Table 1. Comparative analyze

ETL version	Parallel execution	Automatically conversion from Null to Not Null	Automatically conversion from Timestamp to string	Automatically conversion from Timestamp to date
Infosphere Datastage 7.5.3	NO	NO	NO	NO
Infosphere Datastage 8.5	YES	NO	YES	YES

When we import metadata from Oracle to Datastage there isn't 100% compatibility between those metadata types. As showed above, Datastage 7.5.3 doesn't automatically handle even the basic conversion from Date Oracle to date type in Datastage. Instead it see him as a timestamp type attribute. But Datastage 8.5 does the conversion automatically and helps the developer to not make the conversion manually.

I can conclude that inverting in a update for a ETL solution is a good choice, even if this investment must be very well motivated. From my analyze, I can say that migrating from Datastage 7.5.3 to Datastage 8.5 is a step forward because most importantly allows data from sources to be read in parallel by each node of the ETL configuration.

In fact it reduces the processing time by the number of the nodes: *New_Time=Old_Time divided by Number_Of_Processing_Nodes*. I will add the real time features that I will treat in a future article.

References

[1] IBM Redbook; Chuck Ballard, Daniel M. Farrell, Amit Gupta, Carlos Mazuela, Stanislav Vohnik; Dimensional Modeling: In a Business Intelligence Environment, March 2006

[2] Mike Ferguson, "Architecting A Big Data Platform for Analytics", October 2012

[3] IBM Redbook, Advance user guide for Datastage 7.5.3,2006

[4] IBM Redbook, Advance user guide for Datastage 8.5, 2010

Aurelian TITIRISCA (born 1988 in Romania) has graduated the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies in 2010 and also the Faculty of International Relations of the Christian University Dimitrie Cantemir in 2010. He graduated the Scientific Master Program (Economic Informatics) at the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies. Currently, he is pursuing his Ph.D. in the area of big data.