

Data integration approaches using ETL

Alexandra Maria Ioana FLOREA, Vlad DIACONITA, Ramona BOLOGA
Bucharest University of Economic Studies

alexandra.florea@ie.ase.ro, vlad.diaconita@ie.ase.ro, ramona.bologa@ie.ase.ro

Traditional data warehouses and ETL tools have been slowly pushed to expand their limits as big data has become a more and more prominent actor on the analytics stage. This paper analyzes and compares the features of Pentaho Data Integration and Oracle Data Integrator, two of the main data integration platforms. Such tools are relevant in the context of the evolution of the analytics field, which is expanding from classical business intelligence activities to the use and analysis of big data.

Keywords: business intelligence, data warehouse, big data, ETL

1 Introduction

In [1], published in 1958, H.P Luhn tries to define the characteristics of a Business Intelligence System, showing its flexibility in identifying known information, in finding who needs to know it and in disseminating it efficiently either in abstract form or as a complete document. A more modern definition of BIS is given by Howard Dresner: *concepts and methods to improve business decision-making by using fact-based support systems.*

The term business data warehouse was first used in [2], describing an information retrieval and reporting service that runs against a repository of all required business information.

The term Big Data appeared later in connection with volumes of data that are difficult to store, process and analyze using traditional database technologies [3].

There are different opinions regarding the place and purpose of Data Warehouses (DW) and Big Data within an enterprise, the similarities, and differences between those two technologies.

In [4] the authors argue that Big Data provides cheaper solutions to store raw data which usage is not predetermined and where DW is more expensive providing solutions to store cleaned, transformed, and semantically unified data for predetermined usage. Building a

DW within a firm implies a certain level of business integration.

In [5], Bill Immon claims that there is no correlation between a big data solution and a data warehouse. He argues that a data warehouse is an architecture that assures corporate credibility and integrity and Big Data is just a technology for storing data. There are also opinions that state that Big Data is comprised of both technologies and architectures that can be used to extract value from large volumes of different types of data [6]. For example, Hive can provide data warehouse facilities over Hadoop, the best known Big Data ecosystem. It uses HiveQL, an SQL-type language, storing data in a distributed storage filesystem (usually HDFS).

2. Pentaho Data Integration

Pentaho is a powerful Business Intelligence open source suite that offers many features, including reporting, OLAP pivot tables and dash-boarding [7]. The Pentaho engines were developed as community projects and later integrated into the main product. Data integration can be seen as the process that combines data from a variety of sources in order to provide a coherent view. Pentaho Data Integration (PDI) draws its roots from the business intelligence tool Kettle (KDE Extraction, Transformation, and Loading Environment) originally developed for the KDE desktop environment that is mostly found in Linux distributions.

As discussed in [8], ETL is the backbone of the DW architecture, so its performance and quality are relevant for the accuracy, operability, and usability of data warehouses. The data transforming activities can be run in the target database managing system, and the process is sometimes called ELT or in a dedicated environment outside the target (the classic ETL). Many times, the umbrella term of ETL is used no matter where the transformation process takes part. PDI it's not only an ETL tool, but it can also be used in many scenarios such as loading data warehouses or data marts, integrating data in order to have a unified view, migrating data, data cleansing. Spoon is the graphical tool that can be used to design and test every PDI process in the form of transformations and jobs.

These are handled by different parts of the meta-data driven PDI engine. The jobs and transformations can be saved as XML files, in database repositories or in the PDI repository. They are created graphically, using drag and drop, but some steps can be further refined using scripting. Transformations work with streams of data, transforming the rows according to the declared steps. Jobs contain a sequence of transformations and other auxiliary tasks.

In figure 1 we built a mapper and a reducer transformation. In this case, the mapper works on a corpus of text files which it splits into rows, putting a 1 for every word it finds. The output consists of pairs of keys and values in the form of words and counts. In this case, the reducer transformation aggregates the input using the sum function.

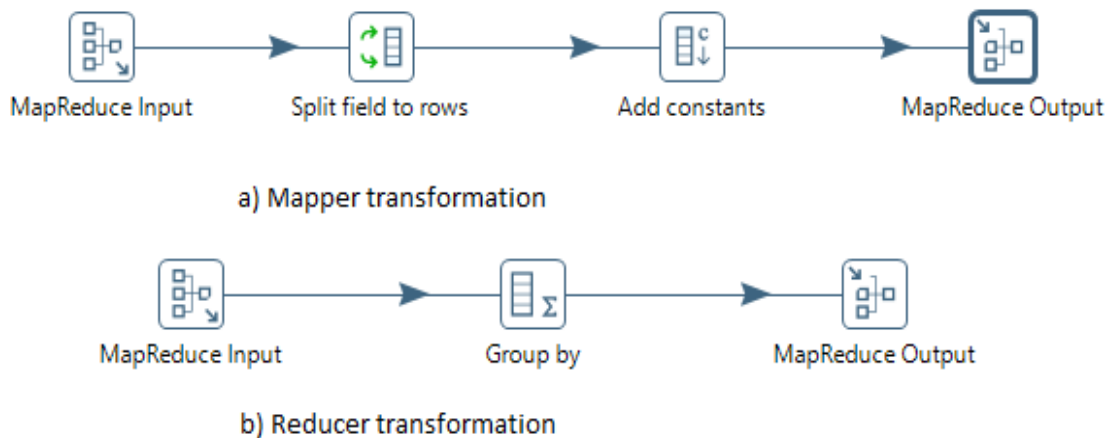


Fig. 1. Mapper and reducer transformations in PDI

In figure 2, we built a job process that connect to a Hadoop cluster, copies a file into HDFS and then runs a MapReduce process. The MapReduce process uses the two prior built transformations, the key

value pairs constructed from the input files by the mapper transformation are after a combination process sent to the reducer processes that sums the appearances of every found word.

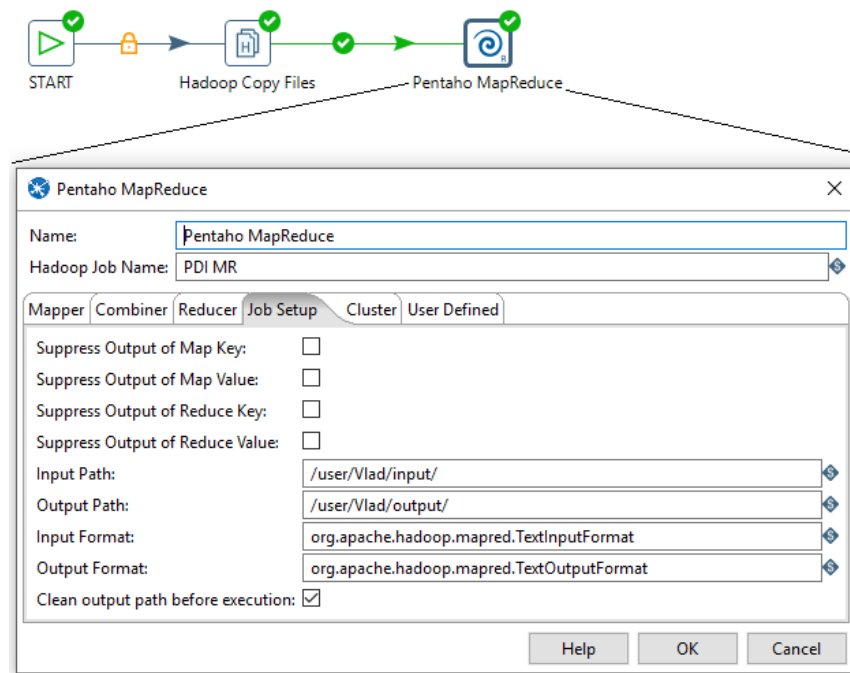


Fig. 2. A PDI job

When the job is executed, Pentaho sends the MapReduce job to the cluster. Its progress can be checked using the cluster's URL of the *Application Tracker*

as shown in figure 3 and the result will be written in the declared Output Path in the cluster's HDFS.

Cluster Metrics

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|----------------------|------------|-----------------|----------------|
| 14 | 0 | 0 | 14 | 0 | 0 B | 2.20 GB | 0 B | 0 | 8 | 0 | 1 | 0 | 0 | 0 | 0 |

Show 20 entries

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress | History |
|--------------------------------|------|-----------------|------------------|---------|-------------------------------|-------------------------------|----------|-------------|----------|---------|
| application_1450864083995_0014 | Vlad | Aggregate | MAPREDUCE | default | Wed, 23 Dec 2015 12:07:57 GMT | Wed, 23 Dec 2015 12:12:25 GMT | FINISHED | SUCCEEDED | | History |
| application_1450864083995_0013 | root | QuasiMonteCarlo | MAPREDUCE | default | Wed, 23 Dec 2015 12:06:15 GMT | Wed, 23 Dec 2015 12:07:06 GMT | FINISHED | SUCCEEDED | | History |
| application_1450864083995_0012 | root | QuasiMonteCarlo | MAPREDUCE | default | Wed, 23 Dec 2015 12:03:47 GMT | Wed, 23 Dec 2015 12:04:44 GMT | FINISHED | SUCCEEDED | | History |

Showing 1 to 3 of 3 entries

Figure 3. Hadoop Application Tracker showing the MapReduce jobs

We can also define jobs and transformations that use a MapReduce approach to run an SQL phrase in order to populate a fact table in Hive, but in our

test environment that proved a very slow task. As shown in figure 4, it took 1h42 minutes to move 319 rows.

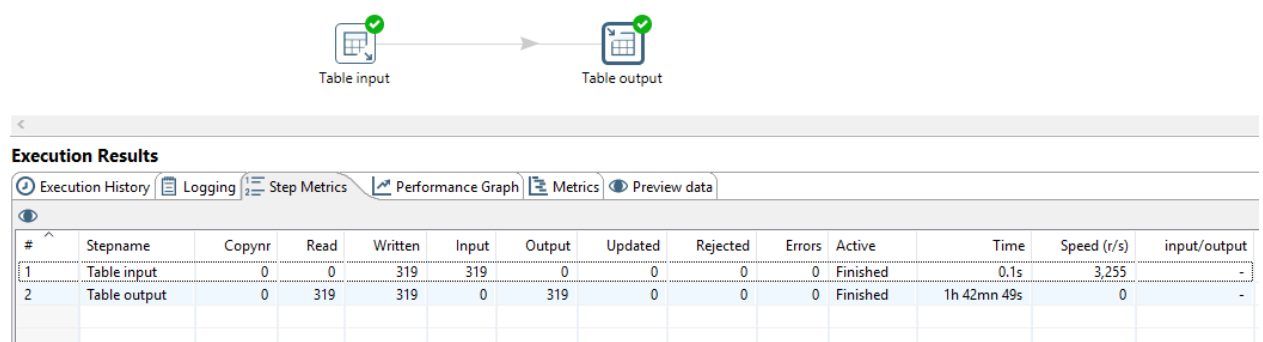


Fig. 4. Loading data from Oracle to Apache Hive using PDI

3. Oracle Data Integrator

As marketed by Oracle, ODI is a highly used, comprehensive platform that covers all of the data integration requirements, starting from high-volume, high-performance batch loads, to event-driven, trickle-feed integration processes, to SOA-enabled data services [15].

It is developed on an unusual architecture that follows the E-LT (Extract-Load-Transform) process instead of the traditional ETL one (Extract-Transform-Load). As a result there is no need for an ETL server situated between the data sources and the target server, the platform using instead the capabilities of the users' RDBMS engine. The traditional ETL process involves downtime of the data warehouse while the loads are performed, but when loading real-time data, there cannot be any downtime of the system and such new solutions must be found. [9]. ODI can perform complex transformations on the source side as well as on the target side, and a large part of these transformations occur in batch mode when there are no end-user queries to be processed by the server.

The data sources for an ODI integration can be extremely complex, varying from different types of data tables to XML files. XML files are regularly used to transmit data from different production units such as power plants [10], [11]. EXtended Markup Language or XML is the basis of all elements which represent the foundation of Web services technology. Considering platform independence, XML is the engine that enables data transfer via the Internet, also constituting the foundation of Web services.[12]

There are five main components of the ODI platform namely the Repository, ODI Studio, the agent, the console and Oracle Enterprise Manager.

The Repository is the central element of ODI's architecture and represents the main storage location where all the information that ODI handles is kept,

namely, connectivity details, metadata, transformation rules and scenarios, generated code, execution logs, and statistics. It consists of two types of repositories, one Master repository which hosts sensitive data such as security and topology information as well as versioned and archived objects and several Work repositories which host project-related data.

ODI Studio represents the GUI of the platform (Graphical User Interface), and it provides access to the repositories to those who use it, regardless if they are administrators, developers or operators. It can be used to administer the infrastructure, reverse-engineer the metadata, develop projects, scheduling, operating and monitoring executions.[13]

The studio is organized in four different Navigators which usually are used by various users according to their roles and security profiles.

The Security navigator is used by system admins and DBAs to manage the roles and privileges of the regular users.

The Topology navigator is also usually utilized by the same type of users as above, in order to define the connections and credentials required for ODI to connect to the source and target systems.

Although developers will use this information most of the time, they do not have the privilege to modify it.

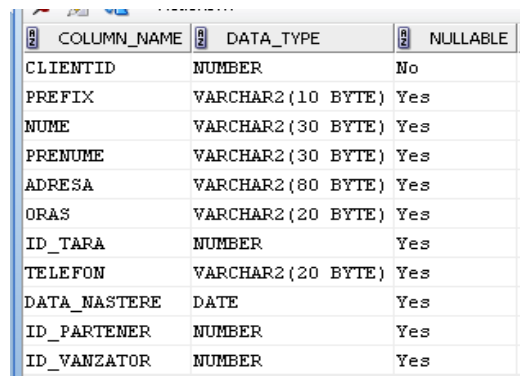
The design navigator is the central part of the developer's world, being used to define the needed transformations in objects called Interfaces.

Operator Navigator is the fourth component of the Studio which ensures the management and monitoring of the activities. Developers use it to check how the code has executed and to debug if necessary.

As presented in [14] the Operator Navigator has the following accordions: session List which displays all sessions organized per date, physical agent, status, keywords, and so forth; hierarchical sessions which display the execution sessions arranged in a hierarchy with their child sessions; load plan - shows the load plan runs of the load plan

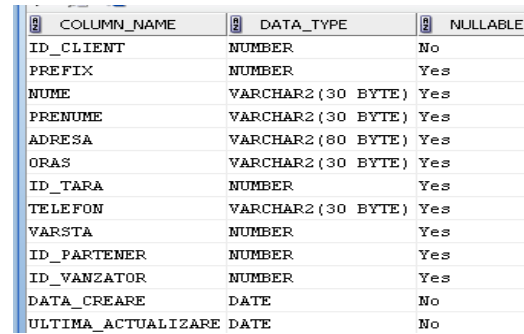
instances; the scheduling accordion shows the list of physical agents and schedules; the scenarios accordion displays the list of scenarios available and the solutions accordion contains the solutions that have been created when working with version management.

In order to demonstrate the capabilities of the operator navigator, we have developed an interface based on the following scenario. We'll be moving data from a Customer System database into a data mart. The source table is called CLIENTI_MASTER (fig. 5) and the target table we will use is the CLIENTI (fig. 6) table located in the DATAMART schema.



| COLUMN_NAME | DATA_TYPE | NULLABLE |
|--------------|-------------------|----------|
| CLIENTID | NUMBER | No |
| PREFIX | VARCHAR2(10 BYTE) | Yes |
| NUME | VARCHAR2(30 BYTE) | Yes |
| PRENUME | VARCHAR2(30 BYTE) | Yes |
| ADRESA | VARCHAR2(80 BYTE) | Yes |
| ORAS | VARCHAR2(20 BYTE) | Yes |
| ID_TARA | NUMBER | Yes |
| TELEFON | VARCHAR2(20 BYTE) | Yes |
| DATA_NASTERE | DATE | Yes |
| ID_PARTENER | NUMBER | Yes |
| ID_VANZATOR | NUMBER | Yes |

Fig. 5. The CLIENTI_MASTER table



| COLUMN_NAME | DATA_TYPE | NULLABLE |
|--------------------|-------------------|----------|
| ID_CLIENT | NUMBER | No |
| PREFIX | NUMBER | Yes |
| NUME | VARCHAR2(30 BYTE) | Yes |
| PRENUME | VARCHAR2(30 BYTE) | Yes |
| ADRESA | VARCHAR2(80 BYTE) | Yes |
| ORAS | VARCHAR2(30 BYTE) | Yes |
| ID_TARA | NUMBER | Yes |
| TELEFON | VARCHAR2(30 BYTE) | Yes |
| VARSTA | NUMBER | Yes |
| ID_PARTENER | NUMBER | Yes |
| ID_VANZATOR | NUMBER | Yes |
| DATA_CREATE | DATE | No |
| ULTIMA_ACTUALIZARE | DATE | No |

Fig. 6. The CLIENTI table

We can easily notice that although the two tables are quite similar there are a couple of differences that need to be addressed through mappings. Most of these will be automated but a couple must be done manually:

- The PREFIX column in our source is a character string, but in the target, it's a number. The Customer System has all the titles normalized to US prefixes, but we may wish to process the data later to have national-specific titles based on country or residence, so our data mart has all prefixes translated into numeric codes.
- Our target stores an age in the 'Varsta' field, but our source has the birth date (data_nastere).

For this, we have built an interface that deals with all the necessary mappings, as shown in figure 7.

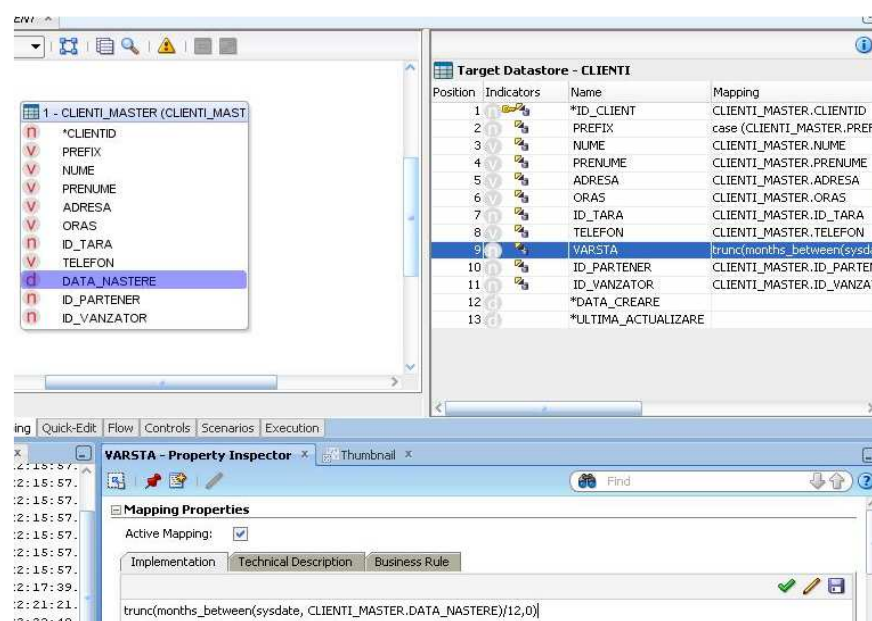


Fig. 7. ODI Interface

In order to verify the results of the interface execution, we use the operator Navigator presented previously.

We can notice that we can visualize the work sessions ordered after several

criteria such as execution agent, session name, session status, the user who executed or we can see all sessions executed, ordered by their number (fig. 8).

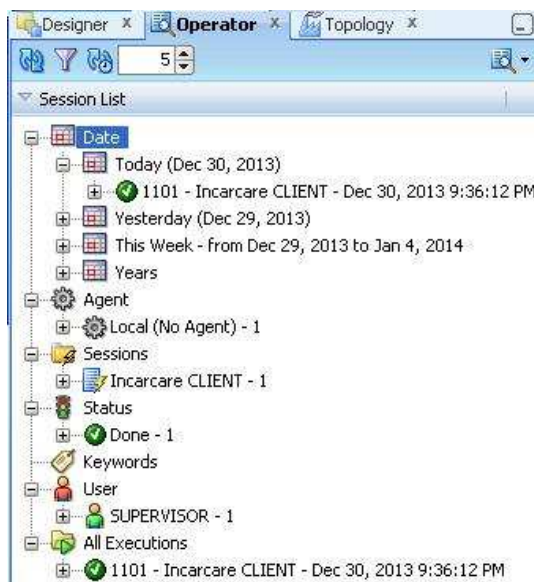


Fig. 8. ODI integration sessions

If we select the 'all executions' node and choose one of the previous sections we will be able to have some general information about that particular execution including duration, the number

of inserts, updates and deletes, and the execution status of the session.

In this particular view, we can see that there have been 3 rows inserted with no updates or deletes (fig. 9).

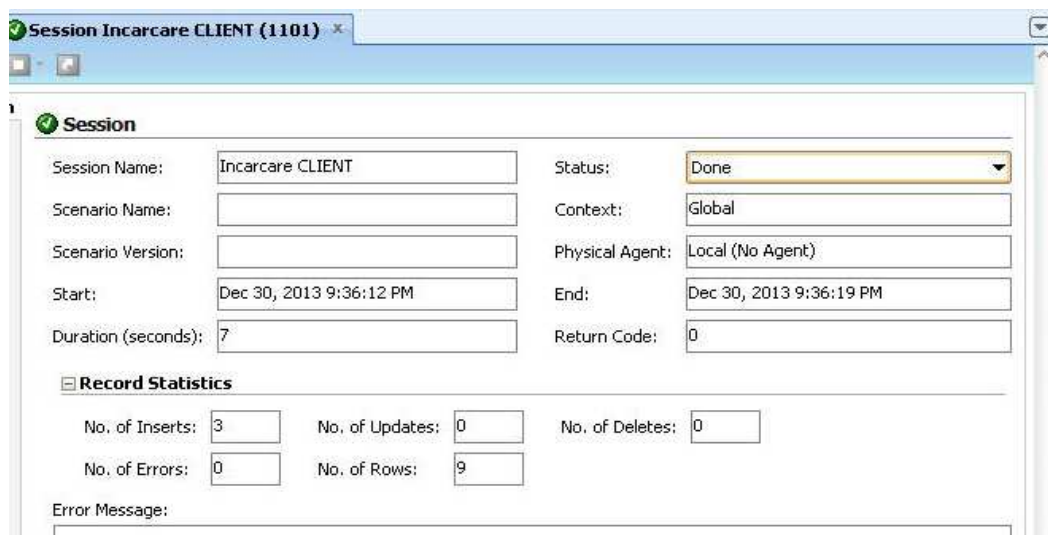


Fig. 9. Overview of an ODI integration session

Back to the list sessions, by expanding the 'Incarcare CLIENT' node, note that the session has an important step, which can then be further expanded to all the

steps ODI created and executed to run the integration interface as seen in figure 10. The steps named 'Loading' have been generated by the loading knowledge

modules (LKM) while the one named 'integration' have been generated by the integration knowledge modules (IKM).

Where there is a yellow triangle, it means that a Warning had been issued, in our case when we tried deleting temporary tables that did not exist at runtime.

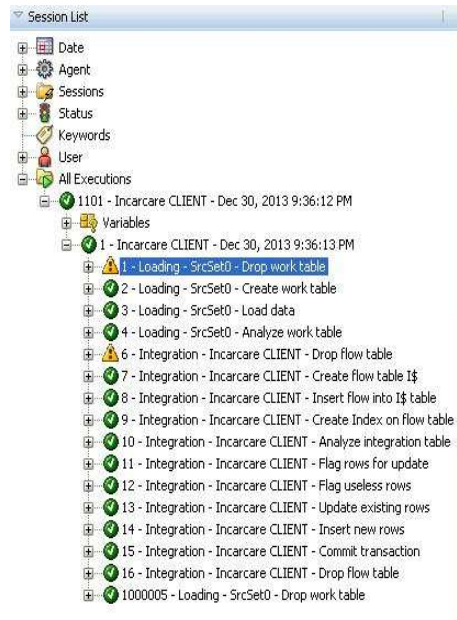


Fig. 10. Sessions' steps

Amongst the 16 steps of the execution, step 3 '*Loading – SrcSet0 – Load data*' time is when the LKM module loads data into a temporary load table in the staging area. We double-click on this step to see information about execution such as before but in this case, the values in the fields for the number of inserts, updates, and deletes refer to the temporary loading work table (C\$) here.

We select the Code tab and view the generated SQL source code for this step and the code correspondent to the destination for this stage (loading the working table in the staging area). We walk through the code and notice the mappings that we've previously specified for the 'age' and 'code' fields are included (fig. 11).

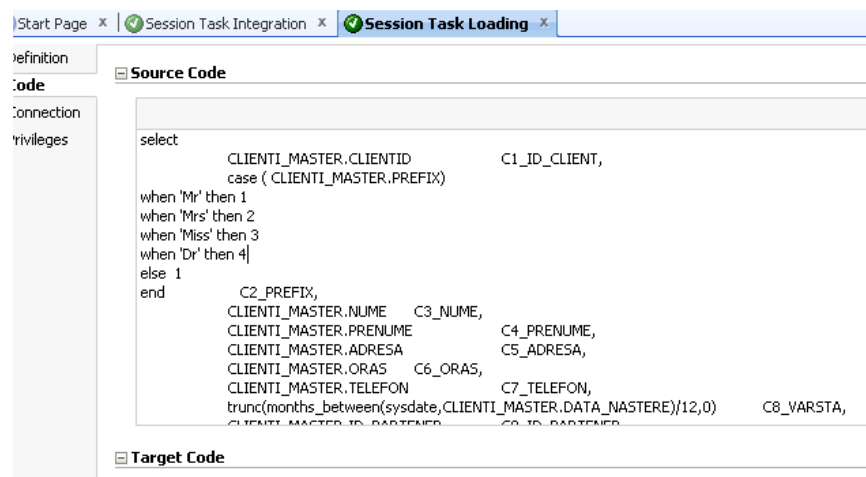


Fig. 11. Mappings information

When we return to viewing the session we choose the session step '*Integration – Incarcare PRODUS – Insert flow into I\$ table*'. The window that opens shows the activity that took place on the temporary integration table. This is the step where the mappings that have been configured to take place in the staging area are executed. In this step runs mappings that have been configured to take place in the staging area.

5 Conclusions

Even though PDI was the first to handle modern data structures and offered connectors to Hadoop clusters, ODI is catching up quickly and is currently offering powerful connectors and Big Data transformations. Compared to ODI, PDI doesn't necessarily need a repository, its jobs and transformations can be saved as XML files. If required, both products can use different database engines (Oracle, Microsoft SQL Server, Sybase, DB2, etc.) to install their repositories. The community, open source version of PDI can be freely downloaded. It's the code can be found on <http://sourceforge.net/>. ODI can be freely downloaded but only for evaluation purposes; the source code cannot be downloaded. We can conclude that both PDI and ODI are powerful, highly customizable integration tools that receive constant upgrades from its developers, so the choice for a particular integration project is more a matter of taste and budget.

References

- [1] H. P. Luhn, "A Business Intelligence System," IBM Journal, 1958. [Online]. Available: <http://altaplana.com/ibmrd0204H.pdf>. [Accessed: 21-Dec-2015].
- [2] B. A. Devlin and P. T. Murphy, "An architecture for a business and information system," IBM Syst. J., vol. 27, no. 1, pp. 60–80, 1988.
- [3] I. A. T. Hashem, I. Yaqoob, N. Badrul Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan, "The rise of 'Big Data' on cloud computing: Review and open research issues," Inf. Syst., vol. 47, pp. 98–115, 2014.
- [4] B. Boulekrouche, N. Jabeur, and Z. Alimazighi, "An intelligent ETL grid-based solution to enable spatial data warehouse deployment in cyber physical system context," 1st YAWL Symp. 2013, vol. 56, no. MobiSPC, pp. 111–118, 2015.
- [5] B. Immon, "Big Data Implementation vs. Data Warehousing," 2013. [Online]. Available: <http://www.b-eye-network.com/view/17017>. [Accessed: 19-Dec-2015].
- [6] R. L. Villars and L. Borovick, "Big Data and the Network," IDC White Paper, 2011. [Online]. Available: <https://www.brocade.com/content/dam/common/documents/content-types/whitepaper/idc-big-data-network.pdf>. [Accessed: 21-Dec-2015].
- [7] R. Bouman and J. Van Dongen, Pentaho® Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL®. 2009.
- [8] A. Karagiannis, P. Vassiliadis, and A. Simitsis, "Scheduling strategies for efficient ETL execution," Inf. Syst., vol. 38, no. 6, pp. 927–945, Sep. 2013.
- [9] J. Popeangă, I. Lungu, "Real-Time Business Intelligence for the Utilities Industry", Database Systems Journal vol. III, no. 4/2012, [Online]. Available: http://www.dbjournal.ro/archive/10/10_2.pdf
- [10] I. Lungu, A. Velicanu, A. Bâra, I. Botha, A. M. Mocanu, A. Tudor – Spatial Databases for Wind Parks, Economic Computation and Economic Cybernetics Studies and Research Journal, ISSN: 0424-267X, nr.2/2012, pp.5-23 [online]. Available: http://www.ecocyb.ase.ro/22012/Lungu%20Ion%20DA_.pdf
- [11] A. Bara, I. Lungu, S.V. Oprea, I. Botha, A. Chinie, "Model assumptions for efficiency of wind power

- plants'operation", Economic Computation and Economic Cybernetics Studies and Research Journal, ISSN: 0424-267X, nr.2, pp.5-23 [online]. Available: [http://www.ecocyb.ase.ro/eng/Articles_4-2014/07%20-%20Ion%20Lungu,%20Adela%20Bara%20\(T\).pdf](http://www.ecocyb.ase.ro/eng/Articles_4-2014/07%20-%20Ion%20Lungu,%20Adela%20Bara%20(T).pdf)
- [12] A. Florea, "Business Process Management Solutions Performance Tuning and Configuration", Database Systems Journal, Vol. II, No. 3/2011, ISSN: 2069 – 3230. Available: http://dbjournal.ro/archive/5/3_Florea.pdf
- [13] Oracle, "Oracle® Fusion Middleware Getting Started with Oracle Data Integrator 12c", 20015, Available: <http://www.oracle.com/technetwork/middleware/data-integrator/overview/odi-12c-getting-started-guide-2032250.pdf>
- [14] P.C. Boyd-Bowman, C. Dupupet, D. Gray, D. Hecksel, J. Testut, B Wheeler, "Getting Started with Oracle Data Integrator 11g: A Hands-On Tutorial", Packt publishing, Mumbai, India, 2012
- [15] Oracle Data Integrator, <http://www.oracle.com/technetwork/middleware/data-integrator/overview/index-0883> (accessed January 06, 2016).



Alexandra Maria Ioana FLOREA obtained her Ph.D. in the field of economic informatics in 2012 and at present, she is a lecturer at the Academy of Economic Science from Bucharest, the Economic Informatics Department. Her fields of interest include integrated information systems, information system analysis and design methodologies and database management systems. She published more than 40 papers in peer-reviewed journals and conference proceedings, many indexed by ISI or SCOPUS and is the co-author of 3 books.



Vlad DIACONITA has a Ph.D. in Statistics and Economic Cybernetics and is a member of the IEEE and INFOREC organizations and member of the technical team of the Database Systems Journal. As part of the research team, he has worked in 4 UEFISCDI funded grants. He has received an award and a scholarship as part of the EU funded Excelis project. He published more than 30 papers in peer-reviewed journals and conference proceedings, many indexed by ISI or SCOPUS. He is the co-author of four books.



Ana-Ramona BOLOGA is an associate professor at the Academy of Economic Studies from Bucharest, Economic Informatics Department. Her Ph.D. paper was entitled "Software Agents Technology in Business Environment". Her fields of interest are: integrated information systems, information system analysis and design methodologies, and software agents.

Copyright of Database Systems Journal is the property of Bucharest Academy of Economic Studies and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.