Prototype of a Web ETL Tool

Matija Novak, Kornelije Rabuzin
Faculty of Organization and Informatics
University of Zagreb
Varazdin, Croatia

***Abstract*— Extract, transform and load (ETL) is a process that makes it possible to extract data from operational data sources, to transform data in the way needed for data warehousing purposes and to load data into a data warehouse (DW). ETL process is the most important part when building the data warehouse. Because the ETL process is a very complex and time consuming, this paper presents a prototype of a web ETL tool that offers step-by-step guidance through the entire process to the end user. This ETL tool is designed as a web application so users can save time (and space) required for installation purposes.**

Keywords—ETL; data warehouse; web; ETL tool

I. Introduction

Databases (DB) have been used for many years and it is hard to imagine any (transaction) application that wouldn’t use some database. Over time people realized that databases, although they support daily operations, are not good source when complex analysis must be made on data. Merging data from multiple tables, the complexity of the model (as such), the inability to generate reports by end users and (in)effectiveness of such approach resulted with the need to reorganize (transform) data into a form that will be suitable for analysis. This form is called a data warehouse [1, p. 85].

The basic idea of data warehouses is to store data in such a way that users can understand and analyze data. R. Kimball and J. Caserta define the data warehouse as follows: "A data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making." [2, p. 23]. This definition says that data warehouses are used to support decision-making. The data warehouse would not be good without the iterative process of extracting, cleaning, conforming and loading data (the so called ETL process) from various sources into the star schema model.

When we talk about data organization in the data warehouse, we distinguish between fact and dimension tables. While dimension tables contain large number of attributes that we use when analyzing (filtering) data, fact tables contain measures to quantify business processes (number of product units sold, number of orders, number and duration of calls, etc.). For end users such model is understandable and they can independently create necessary reports.

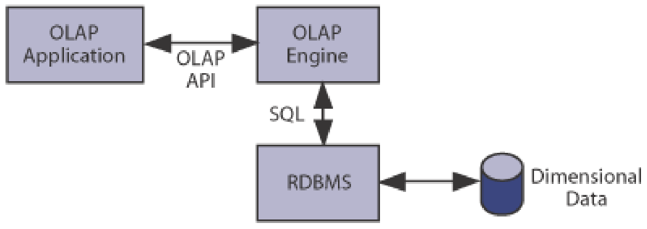


Fig. 1. ROLAP model [3]

Basically there are two mechanisms (ways) that can be used to store data in the data warehouse (Fig. 1): relational online analytical processing (ROLAP) and multidimensional online analytical processing (MOLAP) [4, p. 165] While ROLAP stores data in tables, MOLAP stores data in special structures also known as cubes. There are advantages and disadvantages but we will not discuss them in this paper (more can be found in [9]).

If one looks at today’s market, one can find various Business Intelligence (BI) tools that are used to produce reports by using data from data warehouses. Although data warehouses are very valuable sources of data, the main problem in the construction of the data warehouse is the so called ETL process. Systems for extraction, transformation and loading of data (ETL systems for short) are the foundation of data warehouses. When constructing a data warehouse 70 percent of time and resources is used for the ETL purposes (by Inmon 80 percent [5, p. 295]).

Building a data warehouse is expensive, time consuming and complex job and the ETL phase is the most critical one. Because of that the idea of this paper is to present the ETL tool that should facilitate and accelerate the process of ETL. This ETL tool offers the user step-by-step guidance through the entire ETL process. In addition this ETL tool is designed as a web application and users can save time (and space) required for installation. This tool can start from heterogeneous sources of data and result is a dimensional model stored in a relational database which can be used for other purposes (primarily for building reports by means of some BI tool).

This paper is structured as follows: the second section describes the related work and the third section the basics of ETL. Next, the model of the ETL tool is shown and several screen shoots are given. In the end of the paper some open questions are addressed (future work) and the conclusion is presented.

II. ETL TOOLS

There are various professional tools, which can be used to assist user in the ETL process; however, the problem of these tools is their complexity and/or price. For example, if we take free tool Talend Open Studio, its features are great and user can execute complex operations. But, the tool can be very difficult and confusing, especially if the user is not familiar with the ETL process. Because the tool has a number of possibilities, it is necessary to examine what our individual elements (or rather objects) allow us to do and what are their attributes.

In addition, there is service-oriented architecture (SOA) ETL Framework described in [7] that tries to split the tightly coupled functionalities of an ETL tool into separate parts that can be used as services.

To the authors knowledge there is no such thing as a completely web based tool that would integrate the learning of the ETL process into the tool itself. Furthermore, the ETL tool described in this article is completely web based (it can be easily accessed through web browser, no installation is needed and multiple users can use it at the same time). To avoid problems with the ETL process, the created tool guides the user through the ETL process and teaches him during the way; so the basic idea is that it can be used by people not familiar with the ETL process.

III. ETL

The ETL process is a set of activities that are not visible to the end user and that are taking place in the background. In addition to retrieving information from different sources, many activities need to be performed on data [2, p. xxi]: mistakes have to be corrected, data needs to be structured, etc.

The ETL process (Fig. 2) has three steps [6, p. 139]:

- *Data extraction* – accessing data sources in order to retrieve (required) data.
- *Data transformation* - In this step data collected from various sources is checked, cleaned and conformed, i.e. data undergoes a series of activities in order to improve the quality of data. [4, p. 375]
- *Data loading* - extracted and transformed data is loaded into the data warehouse (dimension and fact tables).

While extraction and loading only transfer data, transformations are really changing data. Kimball and Caserta propose the so-called Extracting, Cleaning, Conforming, and Delivering (ECCD) instead of the ETL, but either way in the end data has to be loaded in the data warehouse. ECCD consists of four steps [2, pp. 18-19]:

- *Extraction* – the first step is to take data from different sources and store it in the ETL environment in order to make the necessary processing.
- *Cleaning* – performing the first transformation of data in order to enhance the quality of the original data.
- *Conforming* – This step is necessary if there are two or more data sources. Various sources tend to have differently shaped and stored data and there is a need to synchronize data (resolve conflicts when different names are used, resolve the problem of duplicates, etc.).

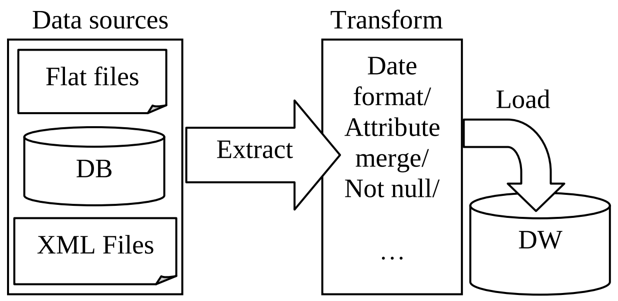


Fig. 2. ETL process steps

- *Delivery* – The last step is the same as for the ETL (loading data into the data warehouse). This process of loading data can be further divided into two parts [2, pp. 161-254]:
    - Loading data into *dimension tables* – they contain information that allows understanding (interpreting) the data in fact tables
    - Loading data into *fact tables* – central tables that contain numerical values.

There is also the fifth step (the so called management) which is not part of the flow of data processing, but it is used for system and process management of the ETL environment. ETL and ECCD describe the same data processing activities and the end result is the same, yet the ECCD is somewhat better because the steps define in detail the activities to be carried out in the processing of the original data and it separates activities related to a single source of data and activities that include multiple data sources. Nevertheless the term ETL is so "domestic" that it is not reasonable to expect that it is replaced in the near future.

A. Metadata

During the ETL process various metadata is generated. The ETL metadata is divided into four main categories [2, pp. 367-368]:

- *ETL job metadata* – is a container of transformations that manipulate the data. Every ETL task is captured here.
- *Transformation metadata* – contains information about every transformation that is used inside of the ETL jobs.
- *Batch metadata* – in the ETL process batches are used to run collections of jobs together. Batches can contain sub-batches and schedules can be made to run batches periodically. All that information is stored in batch metadata.
- *Process metadata* – is generated when batches are executed. Process metadata has information on whether loading of data (into the DW) was successful or not.

1) *Logical data map*

At the beginning of the ETL process, it is necessary to make a logical data map. The logical data map documents the links between the columns (fields) in the source and the columns in the destination table (in the data warehouse). Logical data map is one of the most important and most useful metadata generated by the ETL. Header of the logical data map is shown in the following table [2, pp. 56-71].

Once created, the logical data map provides information about what needs to be extracted, from where, how to process data and where it needs to be saved after processing. The logical data map is useful throughout the entire ETL process.

TABLE I. HEADER OF THE LOGICAL DATA MAP [2, P. 60]

Target					Transformation
Table name	Column name	Data type	Table type	SCD type	
Source					
Database name	Table name	Column name	Data type		

2) *Data sources*

A data warehouse often uses different data sources (Enterprise Resource Planning (ERP) Systems, extensible markup language (XML) files, databases and flat files). No matter which source is used, specific metadata is required. The following metadata attributes are minimally required [2, p. 362]:

- *Database or file system* – "The name commonly used when referring to a source system or file." [2, p. 362]
- *Table specification* – "The ETL team needs to know the purpose of the table, its volume, its primary key and alternate key, and a list of its columns." [2, p. 362]
- *Exception-handling rules* – Necessary information related to the quality of data and how should the ETL process manage them.
- *Business definitions* – It's good to get the business definitions as these two or three sentences are very useful when you need to understand data.
- *Business rules* – "Every table should come with a set of business rules. Business rules are required to understand data and to test for anomalies." [2, p. 362]

Types of data sources can be:

- *Flat Files* - In most data warehouses regular files can't be avoided. Flat files can be used in the ETL system for at least three reasons [2, pp. 90-91]: *delivery of source data*, *working/staging tables* or *preparation for bulk load*. There are two types of files [2, pp. 91-93]: *fixed length flat files* and *delimited flat files.*
- *XML files* - In recent years the XML is used very much. XML files are good for the ETL process because they are self-documented unlike ordinary files that are not. XML files are often used for data exchange and provide independence from the specific computational implementations [4, p. 126].
- *Operational databases* - the most common source of data for the data warehouse. Benefits of databases regarding the ETL phase are [2, pp. 40-41]: *Apparent metadata*, *Relational abilities* (exp. referential integrity), *Open repository* (data can easily be accessed by any structured query language (SQL) compliant tool), *DBA Support* (there is a group responsible for data in database management system (DBMS)), *SQL interface, etc*.
- Other sources:
    - *ERP Systems* – systems that are quite common in organizations.
    - *Master data management (MDM) Systems* - are centralized resources designed to hold the main copy of the key entity, such as a customer or a product.
    - *Web log* – for example a control document that is automatically created from the Web server.

IV. THE MODEL OF THE ETL TOOL

The following figure (Fig. 3) shows the high level architecture of the proposed ETL tool. The user uses web interface to define the metadata (i.e. user creates project, process, group, destination, etc.) that the ETL processing will use. When all data is entered, user runs the thread that extracts information from one source, then performs defined transformation (as necessary) and finally loads data into the data warehouse. After one source is completed, the thread proceeds to the next source. Possible improvement is to implement multithreading in order to process multiple sources at once.

A. *ETL thread*

Data processing is made by the thread that starts after the metadata is entered. Fig. 4 shows the class diagram of this part of the tool. When you start the thread class "Main logic", it is instantiated and it then instantiates classes "Extraction", "Transformation" and "Load". After that the methods of the class "Load" are called to create the destination (dimension and fact tables). Then, the logical data map is read and information is stored into two vectors. The first vector contains metadata relating to data for dimension tables and second vector stores data for fact tables. The thread then moves and processes dimensions, one by one, and SQL query for extraction is created and run. After that, data is transformed as it is described in the metadata entered by the user; after the transformations are done, the loading starts to load data into the data warehouse (row by row). When dimensions are finished, the fact tables are processed in the next step (the procedure is the same but one has to have in mind that fact tables have to be connected to specific dimension tables).

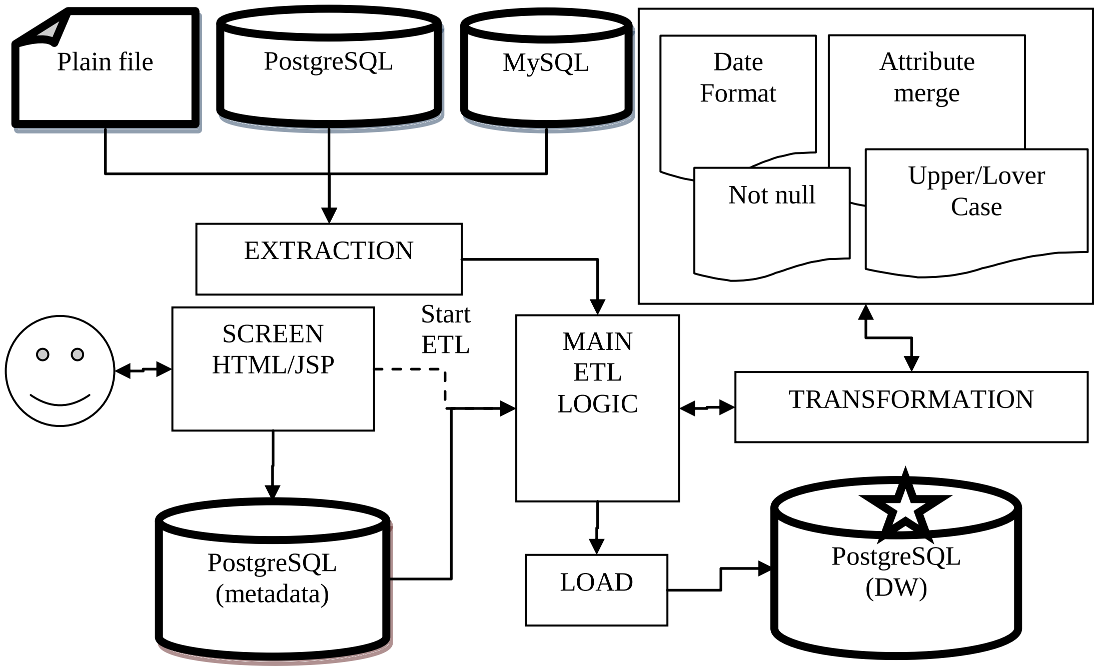


Fig. 3. ETL tool High-level architecture

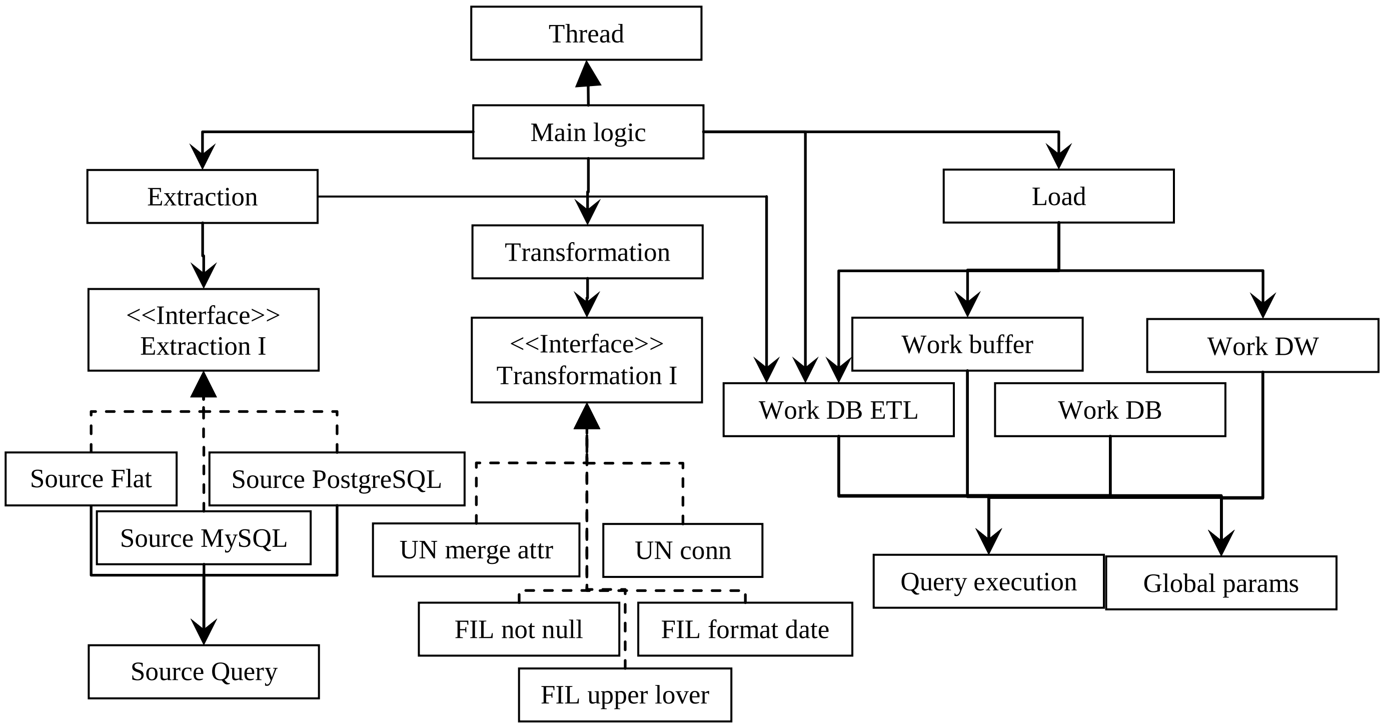


Fig. 4. Class diagram – ETL thread

B. *Dynamic loading*

In order to create a flexible tool and have the option of upgrading, dynamic load of classes and JSP files has been implemented in two places:

- *In source extraction part* – for every source type one class has been made;
- *In data transformation part* - for every transformation that the tool can perform one class has been made;

Since each type of source and each transformation have their own class, it is possible to add new types of sources or new transformations. All you need to do is create a class (and if needed a JSP file) and add metadata info about it.

Three important things enable dynamic loading of classes:

- Each class of source type (or transformation) must have a method that returns an instance of a class within the class itself (Fig.5).
- The interfaces implemented by source type or transformation classes (Fig. 6).
- The class that has methods to search for required class through its name and dynamic load of the class into memory and methods to search for functions within the retrieved class that return an instance of the desired class (Fig. 7)

In addition to the dynamic class loading, transformations also use dynamic load of JSP files which contain fields (if necessary) that the user must fill in when choosing this particular transformation. JSP loading is done with AJAX.

C. *Tool configuration*

In order to use the ETL tool, administrator has to pre-configure it. Most important are the following parts:

- *Source types* (Fig. 8) – it refers to the source types that the tool can work with (for now PostgreSQL, MySQL and flat file with delimiter)

```
public class Source_MySQL
 implements Extraction_I {
  public static Source_MySql
           get_instance(String args[]){
    Source_MySQL instance =
                    new Source_MySQL();
    return instance;
  }
  public boolean load_parameters(
    String address, String name,
    int port, String username,
    String password){...}
  public Vector get_table_columns(){...}
  public Vector execute_query(
   String query, Vector info){...}
}
```

Fig. 5. Example 1 Example of dynamic loading class

```
public interface Extraction_I {
  public boolean load_parameters(
    String address, String name,
    int port, String username,
    String password);
  public Vector get_table_columns();
  public Vector execute_query(
    String query, Vector info);
}
```

Fig. 6. Example 2 Example of the interface that dynamic loading class must implement

```
public class Extraction {
  private Extraction_I extraction_i;
  public boolean set_class_instance(
   String src_class) {
    Thread t = Thread.currentThread();
    ClassLoader c =
               t.getContextClassLoader();
    Class toRun = null;
    try{toRun = c.loadClass("subsys_ext."
                        +src_class); ...}
    Method mainMethod = null;
    try{mainMethod =
        findMain(toRun,"get_instance");
        }...
        Object instance = null;
        try{ instance =
              mainMethod.invoke(null, new
               Object[]{new String[1]});
            }...
        extraction_i =
                (Extraction_I) instance;
        return true;
  }
  private Method findMain(
  Class my_class, String function_name) {
    Method[] methods =
              my_class.getMethods();
    for (int i = 0; i < methods.length;
         i++) {
          if (methods[i].getName()
              .equals(function_name))
                 return methods[i];
          }
          return null;
    }}
  public void some_method()
  {...
    extraction_i.load_parmeters(address,
filename, port, username, password);...}}
```

Fig. 7. Example 3 Example of a class that dynamically loads another class [8, p. 11]

- *Transformations* (Fig. 9) – defines which transformations does the tool support, defines the names of classes that implement some particular functionality and the corresponding JSP file which is loaded when the user chooses this transformation.
- *Checkpoints or steps* (Fig. 10) – administrator has to define steps that user follows when filling in the metadata (the administrator must define the page (a JSP file) that opens when user is on a particular step as well as the checkpoints name);

As we mentioned earlier, the program guides the user through the entire process. Fig. 10 shows the steps (checkpoints) for the user; the user has to define how much sources are going to be used.

After that (Fig. 11) we see the input form that is used to define a new data source (there is new PostgreSQL source defined). It is always possible to choose from already existing sources. The tool will use that info and will connect to the source and will retrieve metadata as well. When we have all (sources) metadata and we have defined dimension and fact tables with attributes, the user must define all merges of the attributes (Fig. 12) (for example merge of first and last names into the attribute buff_name_surname).

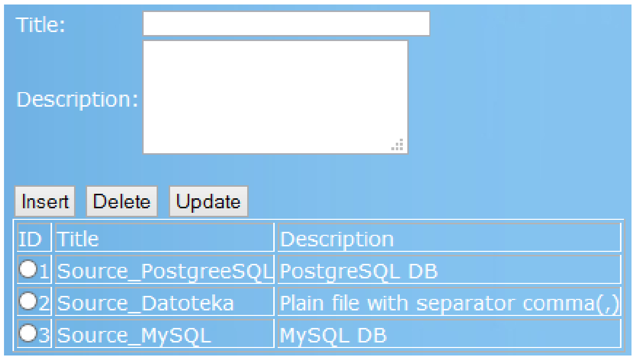
After this is defined, the user can connect the attributes from the source to destination attributes and define transformations that need to be done. When this step is done for all dimensions/fact tables and the corresponding sources, the last step starts the thread for ETL processing. Before starting the thread the user can change entered data and go back to previous steps.

V. Conclusion

The ETL process is the most important and most problematic part when creating data warehouses. In order to speed up the whole process and in order to make it easier (for users), we built a tool that leads the users through the whole process. Although this ETL tool is far from being perfect and cannot be measured with professional tools on the market, its major advantage is that it is web based; no installation is needed, it is available right away, more users can use it at the same time and users can learn the ETL process when using the tool. The ETL tool is good for users who are not that familiar with the ETL process and who have no time to analyze new ETL tools but want to summarize data, move data into the data warehouse and analyze data. The ETL tool is flexible and because of that it can be easily upgraded.

VI. Future Works

Because this tool is only a prototype, there are many possible improvements. Some parts are already improved; some complex queries were made that extracted more data at once, some filters were implemented to retrieve relevant data (to speed up the tool) etc. In the future we plan to optimize the tool (speed, design, source code, DB queries, security), add new features (add new data sources, new transformations, etc.) and test the tool with larger set of data and compare results to other tools. Also, it is planned to take data from two grocery stores (data from a small data warehouse that was implemented a few years ago) and test the ETL tool with that data and compare it first to manual ETL, and later with other tools. When this is done and tool is optimized, it is planned to do a research with experts where experts should give feedback about usage of the tool in comparison to the tools they are using right now.



ID	Title	Description
1	Source_PostgreeSQL	PostgreSQL DB
2	Source_Datoteka	Plain file with separator comma(,)
3	Source_MySQL	MySQL DB

Fig. 8. Administration view of source types

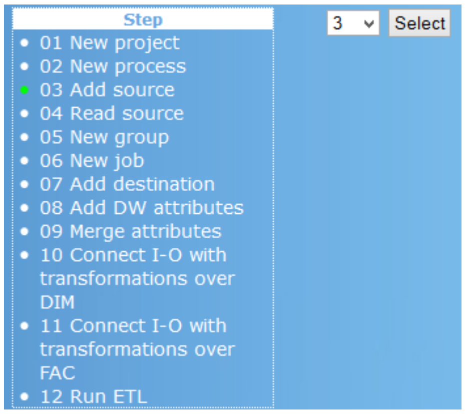
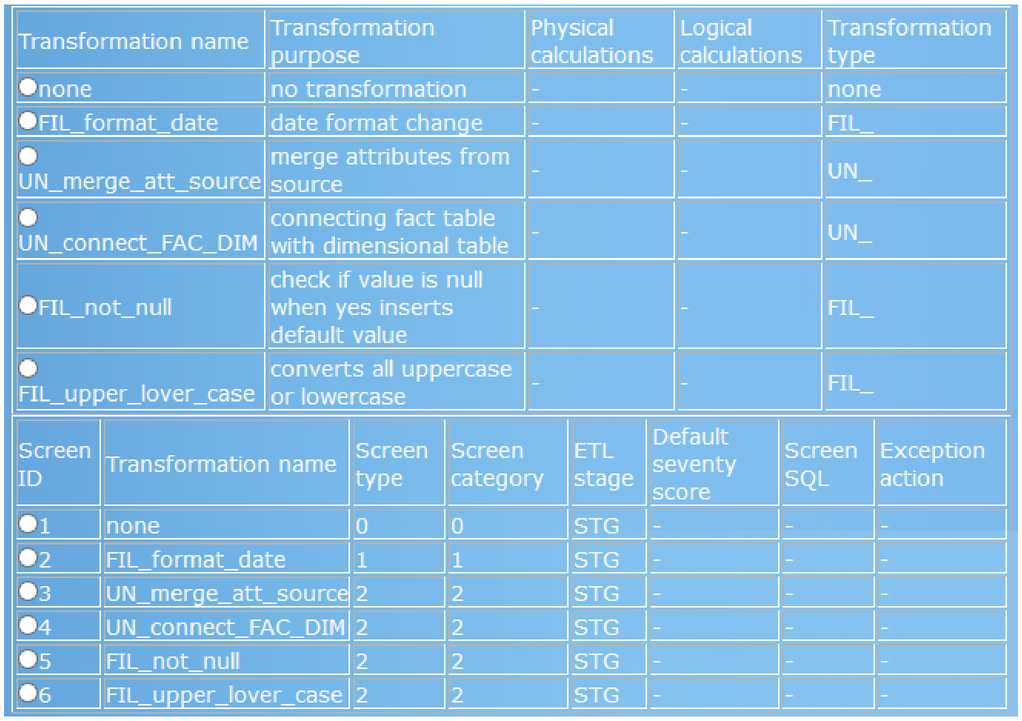


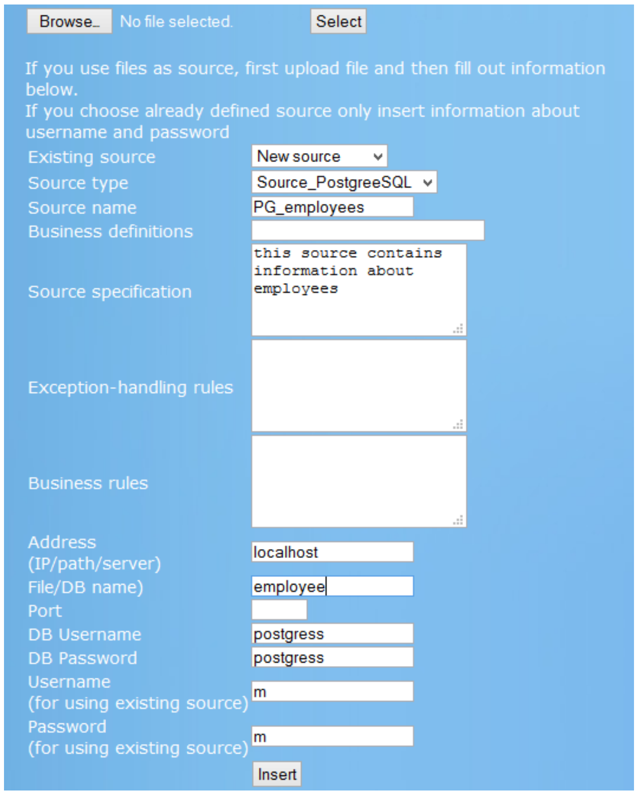
Fig. 9. Menu of checkpoint (steps) for the user (left) and form to select number of sources (right)



Transformation name	Transformation purpose	Physical calculations	Logical calculations	Transformation type
○none	no transformation	-	-	none
○FIL_format_date	date format change	-	-	FIL_
○ UN_merge_att_source	merge attributes from source	-	-	UN_
○ UN_connect_FAC_DIM	connecting fact table with dimensional table	-	-	UN_
○FIL_not_null	check if value is null when yes inserts default value	-	-	FIL_
○ FIL_upper_lover_case	converts all uppercase or lowercase	-	-	FIL_

Screen ID	Transformation name	Screen type	Screen category	ETL stage	Default seventy score	Screen SQL	Exception action
○1	none	0	0	STG	-	-	-
○2	FIL_format_date	1	1	STG	-	-	-
○3	UN_merge_att_source	2	2	STG	-	-	-
○4	UN_connect_FAC_DIM	2	2	STG	-	-	-
○5	FIL_not_null	2	2	STG	-	-	-
○6	FIL_upper_lover_case	2	2	STG	-	-	-

Fig. 10. Administration view of existing transformations and corresponding screen dimension



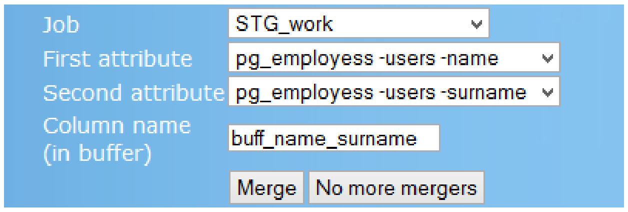
Browse... No file selected. Select

If you use files as source, first upload file and then fill out information below.
If you choose already defined source only insert information about username and password

Existing source	New source
Source type	Source_PostgreeSQL
Source name	PG_employees
Business definitions	
Source specification	this source contains information about employees
Exception-handling rules	
Business rules	
Address (IP/path/server)	localhost
File/DB name)	employee
Port	
DB Username	postgress
DB Password	postgress
Username (for using existing source)	m
Password (for using existing source)	m

Insert

Fig. 11. Form for entering new source



Job	STG_work
First attribute	pg_employess -users -name
Second attribute	pg_employess -users -surname
Column name (in buffer)	buff_name_surname

Merge No more mergers

Fig. 12. Form to define attribute merges

REFERENCES

[1] K. Rabuzin and M. Novak, "Data warehouses and ETL," Methods and Tools for Information and Business Systems development (Case22), Zagreb, Jun. 2010, pp. 85-89

[2] R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data, Indianapolis: Wiley Publishing Inc., 2004.

[3] C. White: "OLAP in the Database - Intelligent Business Strategies" June 2003. http://www.information-management.com/issues/20030601/6807-1.html?pg=2. [Accessed 3 August 2010].

[4] R. Kimball R., M. Ross, W. Thornthwaite, J. Mund and B. Becker, The Data Warehouse Lifecycle Toolkit – Second Edition, Indianapolis: Wiley Publishing, Inc., 2008.

[5] H. W. Inmon, Building the Data Warehouse – Third Edition, New York: John Wiley & Sons Inc., 2002.

[6] F. Silvers, Building and Maintaining a Data Warehouse, Boca Raton: CRC Press, 2008.

[7] I. M. M. Awad, S. M. Abdullah and M. A. B. Ali, "Extending ETL framework using service oriented architecture", Procedia Computer Science, vol. 3, 2011., pp. 110-114

[8] T. Neward, "Understanding Class.forName - Loading Classes Dynamically from within Extensions" 2000. http://media.techtarget.com/tss/static/articles/content/dm_classForname/DynLoad.pdf. [Accessed 5 July 2010].

[9] P. Ponniah, Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals, New York: John Wiley & Sons Inc., 2001.