WebETL Tool – A Prototype in Action

Kornelije Rabuzin, Matija Novak
Faculty of Organization and Informatics
University of Zagreb
Varazdin, Croatia
{kornelije.rabuzin, matija.novak}@foi.hr

***Abstract* – Every grocery store has data about bills, bought items, etc., usually stored in a database (DB). There is often a need to analyze this data. This is not suitable to do on an operational DB, especially when data from two or more stores must be analyzed. Therefore, data from the operational DB is extracted, transformed, and loaded (ETL) into a data warehouse (DW). The ETL process is the most important part of building a data warehouse. It is used to extract data from operational data sources, transform the data (as needed) and to load the data into a data warehouse. Because the ETL process (as such) is very complex and time consuming, a prototype of a web ETL tool was built and tested. The results have shown that it is possible to build a completely web-based ETL tool and that it is faster than manual ETL. Also, because it is completely web based, multiple users can use the tool at the same time, with no installation is needed.**

Keywords – ETL; data warehouse; web; ETL tool

I. Introduction

Data warehouses are used to store data in a so-called star schema (Fig. 1) that consists of dimensional and fact tables. This model is understandable to the end user, thus making it easier to perform data analyses. [1, p.85]

Although many different definitions can be found; according to Kimball & Caserta, data warehouse is defined 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]

The most important part of a DW system is the ETL process, sometimes called extract, clean, conform, and delivery (ECCD) process, as described by Kimball & Caserta in [2, pp. 18-19]. During the construction of a DW, by Inmon [3, p. 295], 80 percent of the time and resources are consumed by the ETL process.

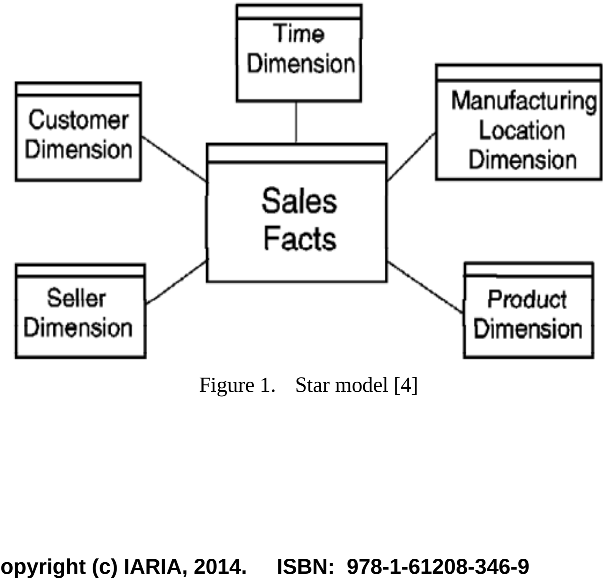


Figure 1. Star model [4]

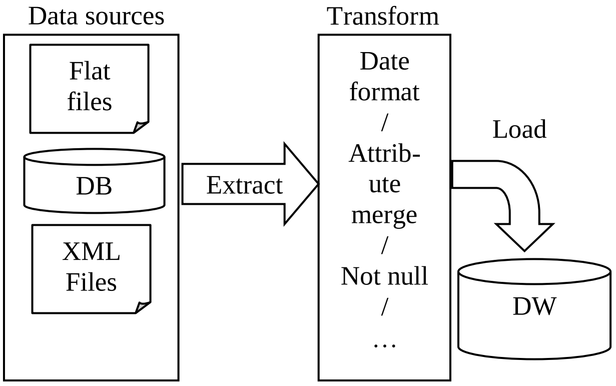


Figure 2. ETL process steps

The ETL process has three important steps (Fig. 2):

- Extract - The first step, extracts data from operational sources such as flat files, operational databases, extensible markup language (XML) files, Enterprise Resource Planning (ERP) systems, etc.
- Transform (clean and conform in ECCD) – The second step, transforms data (e.g., date format, capitalize names, merge attributes, etc.) and the main purpose of this step is to enhance the quality of the original data. In addition, this step resolves conflicts (e.g., duplicate values) if more than one data source is used.
- Load (delivery in ECCD) – The third step, loads transformed data into a data warehouse.

To ease the ETL process, tools are built to support the ETL process. Existing tools are desktop tools, which means they require time for installation and configuration, space to store data created during usage of the tool. In addition, these tools are very complex so user must be very familiar with ETL process and learn how to use the tool. All that requires some time before the tool is efficiently used.

To support the ETL process, and remove the above-mentioned problems, a completely web-based ETL tool was built. "This ETL tool is designed as a web application so that users can save time (and space) required for installation purposes."[5] To help users during the ETL process, step-by-step guidance throughout the entire process was built into the tool making the whole process much easier, even for less experienced users. The explanation on how to configure and use the tool is described in another article [5] that has been submitted for publication. This article is focused on testing the prototype on a real case scenario.

This paper is structured as follows: In Section 2, the ETL tool basics are described. Section 3 describes the model (a real data warehouse example that we were working on). In Section 4, the test results are presented. Section 5 concludes and presents some open questions, listed for future research.

II. ETL TOOL DESCRIPTION

Since many professional ETL tools exist on the market, it is reasonable to ask why should another tool be used and implemented. The main difference between existing tools and our tool is that our ETL tool is completely web-based.

The main advantages of using a web-based tool are:

- No installation needed.
- More users can use it at the same time.
- Lower maintenance costs, because users do not need hardware like in traditional client ETL application, only a browser is needed.

In the article [6], authors introduce web-based the ETL framework and describe the benefits of using a web-based ETL tool, such as lower maintenance costs.

The main difference between the tool presented in [6], is that our tool implements the ETL process in such a way that users can learn the process as they go (the tool guides a user through the steps (Fig. 3) that have to be carried out in the ETL process). Steps are visible throughout the entire process, so the user at every moment knows where he is. During the process, only the right side of the tool changes, as presented in other snapshots (Figs. 4-6 and Fig. 8 and Fig. 9). Therefore, the main two advantages are as follows:

- User can be a beginner in ETL and still be able to use the tool correctly and learn the ETL process while using it.
- Guidance while using the tool through steps (Fig. 3).

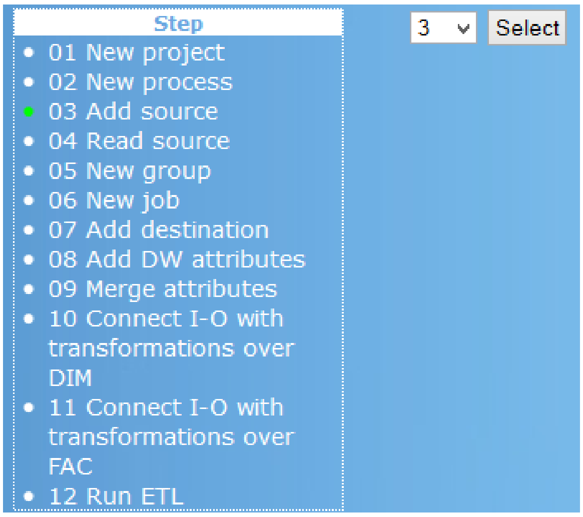
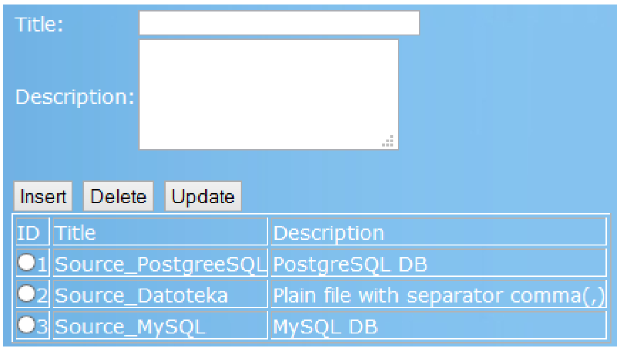


Figure 3. Menu of checkpoint (steps) for the user (left) and form to select number of sources (right)



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

Figure 4. Administration view of source types

The tool is implemented in Java and, for the interface, Hypertext Markup Language (HTML) and JavaServer pages (JSP) sites are used. In order to use the tool, a user has to enter all required metadata. A thread will run and perform extraction, transformation and load of data into a data warehouse. The tool has just one main thread that does the management of three parts: extraction, transform and load. A detailed explanation of the tool is given in [5].

Currently, the tool supports three types of data sources: *Flat source files*, *PostgreSQL* and *MySQL*. However, the tool can be easily expanded to other data sources. The sources that will be used can be located anywhere on the web. The only requirement is an open connection, so that the tool can access the source over Internet.

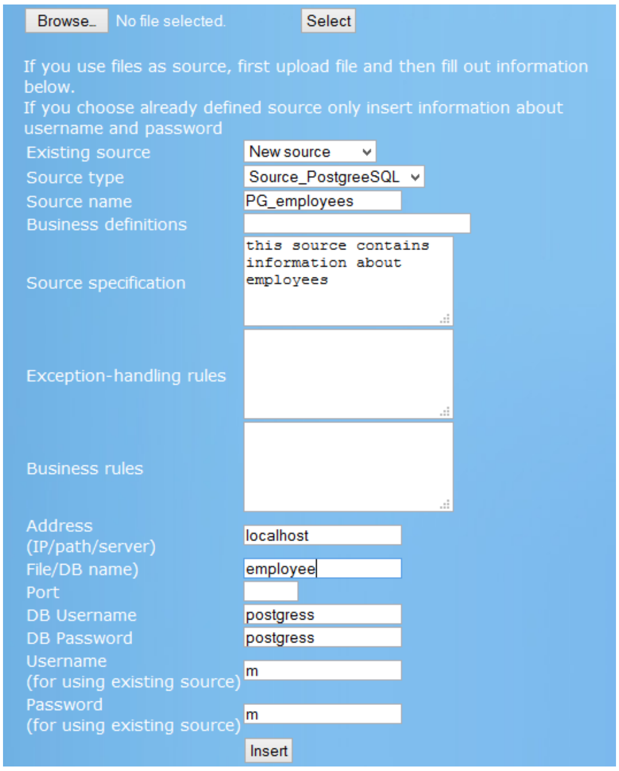
The tool supports extraction from multiple sources into the data warehouse while removing the duplicates based on the unique key (key can consist of one or more attributes) defined for each dimension.

At this point, four types of transformations are implemented; one can:

- Add default value if the source value is empty.
- Merge two (or more) attributes (e.g., name and surname into “name surname”).
- Change date format (change the date and/or time format).
- Change all letters to upper or lower case.

To support the flexible ETL tool, data source and transformation parts were made with dynamic loading. It is possible to add new classes for new data source types as well as new transformations at any time without editing the source code (in order to add new classes, those classes must implement a specific interface). Information about these new classes must be entered on the administration site (Fig. 4). Title in Fig. 4 is the name of the class that implements the functionality for one data source.

As already mentioned, the whole process is guided (step-by-step). Through the guided interface (Fig. 3) the user is asked to enter the information about data sources (Fig. 5, e.g., Internet protocol (IP) address, username, password, DB name). Once all data sources are defined (Fig. 5), the tool reads metadata info about tables and attributes in defined sources (for example, in MySQL, the “information_schema” table). Then, one has to define dimensional and fact tables (with their attributes). Next, the user must match (Fig. 6) attributes from the data source with attributes in the destination, or fact, tables, and define transformations that must be made.



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

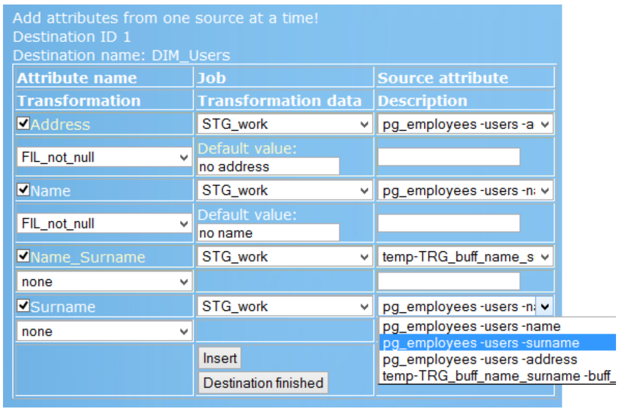
Figure 5. Form for entering new source

This transformation metadata vital and it is stored in the logical data map.

The logical data map is stored as a table that has three main parts [2, pp. 56-71]:

- Destination – described by table name, column name, data type, table type and slow changing dimension (SCD) type
- Source – described by database name, table name, column name and data type
- Transformation – information about what transformations are performed on a specific attribute

Based on all of the metadata, the ETL process can start. One by one, dimensions are processed and for each dimension, data sources are read. When all dimensions are processed, fact tables are processed and connected to corresponding dimensions as defined in metadata.


Add attributes from one source at a time!
Destination ID 1
Destination name: DIM_Users

Attribute name	Job	Source attribute
Transformation	Transformation data	Description
☑Address	STG_work	pg_employees -users -a
FIL_not_null	Default value: no address	
☑Name	STG_work	pg_employees -users -n:
FIL_not_null	Default value: no name	
☑Name_Surname	STG_work	temp-TRG_buff_name_s
none		
☑Surname	STG_work	pg_employees -users -n:
none		pg_employees -users -name pg_employees -users -surname pg_employees -users -address temp-TRG_buff_name_surname -buff_
	Insert Destination finished	

Figure 6. Attribute matching and transformation define

Because data in the data warehouse is stored in relational tables, this mechanism is called relational online analytical processing (ROLAP). There is another mechanism called multidimensional online analytical processing (MOLAP). More about these two mechanisms can be found in [7].

The next Section will show how was test done and what the results are.

III. Test and Results

For testing purposes, we used data from a small data warehouse that we implemented a few years ago (more precisely, data from two grocery stores). Every store had its own operational database and data from those two systems were integrated in order to analyze sales results. This project was implemented in a Microsoft (MS) Access (database management system) and Business Objects XI (business intelligence tool). For testing purposes, we decided to implement the web based ETL tool.

In order to analyze sales results, we had to analyze databases that were used to store data. In each store, the following tables were important for us:

- *Articles* - contains information about products (article name, retail price, purchase price, code, units of measure, etc.),
- *Salesman* – this table contained data on sales people (name, surname, id …),
- *Bills* – contains data about bills (date and time when the bill was made, id of salesman that made it, bill number, payment method, etc.), and
- *Bought items* - contains data about sold items (bill number, article name (connection to article table, code was not used), quantity, retail price, unit of measure, amount, purchase price, tax …).

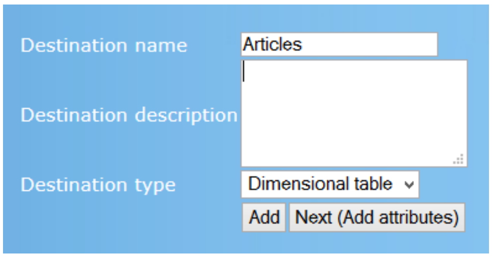
Because the original source was a very old MS disk operating system (DOS) application, MS access was the easiest way to extract data. Although the original data sources had a little less than 500 000 records, for the purpose of this research, we took about 10% of data in this initial testing stage.

```
INSERT INTO sales (bill_number, date_time, article, salesman, item_amount, quantity, profit)

SELECT bills1.bill_number, dates1.id, articles1.code, salesmans1.id, bill_items1.amount, bill_items1.quantity, IIf(cint([bill_items1.tax])=0, [bill_items1].[amount] -[bill_items1].[purchase_price], ([bill_items1].[amount]-1.22*[bill_items1].[purchase_price])/1.22) AS Expr1

FROM salesman INNER JOIN (salesmans1 INNER JOIN (articles1 INNER JOIN (article_helpers1 INNER JOIN ((bills1 INNER JOIN bill_items1 ON bills1.bill_number = bill_items1.bill_nubmer) INNER JOIN dates1 ON bills1.date = dates1.date) ON article_helpers1.article = bill_items1.article) ON articles1.id = article_helpers1.id) ON salesmans1.id = bills1.salesman_id;
```

Figure 7. Example 1



Destination name	Articles
Destination description	
Destination type	Dimensional table

Figure 8. Dimension crate

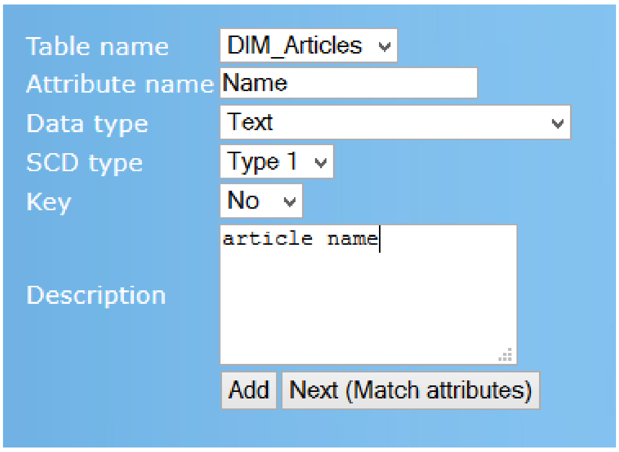


Table name	DIM_Articles
Attribute name	Name
Data type	Text
SCD type	Type 1
Key	No
Description	article name

Figure 9. Destination attribute create

Because these two stores were at different locations, they both had information about articles; the problem was that some articles were equally tagged (the code was the same), but the article name was different. Further, some articles were only present in one store and some were present in another store. When we implemented the ETL manually, we merged these two tables and we created a new dimension table with a new id value. Of course, we had to maintain a connection to bills and items sold in order to load the data into a fact table. A query that was used to load data into a fact table is shown below (Fig. 7).

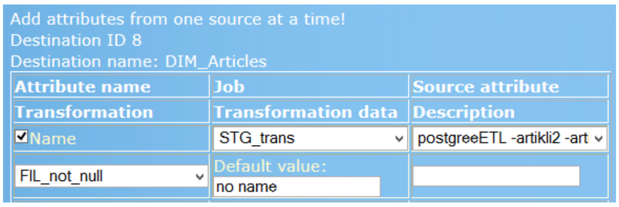
Of course, many other things had to be resolved as well; this was just an example of some transformations that we had to carry out.

In order to test the ETL tool, some data (from two grocery stores) were placed in a MySQL 5.6.11 database and some data was placed in PostgreSQL 9.3 database. For DW, ROLAP model is used; the dimension and fact tables are stored in PostgreSQL 9.3 in star schema format.

As described in previous Section, information of these two sources (Fig. 5) was entered in the ETL tool. We performed the ETL graphically, so no structured query language (SQL) queries needed to be written; the tool generated all queries automatically.

In the first part, we defined data sources (Fig. 5), dimension tables (Fig. 8), dimension table attributes (Fig. 9), fact table, and fact table attributes:

- Dimension Articles (id, article name, retail price, purchase price, code, units of measure, etc.)
- Dimension Salesman (id, name and surname, etc.)
- Dimension Date (id, date, time, date_time, etc.)
- Fact Sales (id, bill number, quantity, dim articles id, dim salesman id, dim date id)



Add attributes from one source at a time!
Destination ID 8
Destination name: DIM_Articles

Attribute name	Job	Source attribute
Transformation	Transformation data	Description
☑ Name	STG_trans	postgreeETL -artikli2 -art
FIL_not_null	Default value: no name	

Figure 10. Attribute “name”, matching and transformation define from one source

```
SELECT bill_items1.quanity as quanity,
bill_items1.ammount as amount, bills1.bill_number as
bill_number, bills1.date as date, bills1.time as time,
articles1.code as code, salesmans1.saleman as sales-
man from bill_items1, bills1, articles1, salesmans1
where bills1.bill_nubmer = bill_items1.bill_number
and articles1.article = bill_itmes1.article and
bills1.salesman_id = salesmans1.salesman_id
```

Figure 11. Example 2

Then, we defined transformations for attributes, example (Fig. 10) for attribute “name” in dimension table “DIM_articles” read from PostgreSQL source table “Article” it was defined transformation “Fill_not_null” and default value was added “no name”. In this way, all attributes were specified from both sources MySQL and PostgreSQL.

Another example, as shown in Fig. 6, is for one dimension (Salesman) and one source (PostgreSQL). To have the “Name_Surname” field in salesman dimension, the merge transformation was used, and for date dimension attributes the “date_format” transformation.

When all metadata was defined, we started the process. First, based on entered data, extraction was done for every defined source (in our case two sources MySQL and PostgreSQL). Second, data was transformed based on chosen transformations; and finally it was loaded into dimensional tables. When all dimension tables were loaded, fact table must be loaded as well.

TABLE I. SOURCE DATA TABLES AND ATTRIBUTES

Table	Attribute	Value
Articles	Article name	White Bread
	Retail price	8,80
	Unit of measure	Item
	Purchase price	6,92
	Id	38503177
Salesman	Id	01
	Name	John
	Surname	Doe
Bils	Bill Number	33703
	Date	2008-07-07
	Time	19:39
	Amount	8,80
	Salesman Id	01
	Payment method	Cache
Bought items	Article Name	White Bread
	Quantity	1
	Retail price	8,80
	Unit of measure	Item
	Amount	8,80
	Purchase price	6,92
	Tax	22
	Bill number	33703

ISBN: 978-1-61208-346-9

```
INSERT INTO FAC_sales
(quantiy, amount, bill_number, id_DIM_date,
id_DIM_Articles, id_DIM_salesmans )
VALUES(1.0,8.80,33703.0,4653,473,3);
```

Figure 12. Example 3

TABLE II. DIMENSIONAL AND FACT TABLES AND ATTRIBUTES

Table	Attribute	Value
DIM_Date	ID	4653
	Date	2008-07-07
	Time	19:39:00
	Date Time	2008-07-07 19:39:00
DIM_Users	ID	3
	Name	John
	Surname	Doe
	Name Surname	John Doe
	Salesman Id	01
DIM_Articles	ID	473
	Name	White Bread
	Retail price	8,80
	Unit of measure	Item
	Purchase price	6,92
	Article Id	38503177
FAC_Sales	ID	18874
	Id DIM Users	3
	Id DIM Articles	473
	Id DIM Date	4653
	Quantity	1
	Amount	8,8
	Bill number	33703

Example 2 (Fig. 11) shows the generated query that extracted data for fact table. The query presented in Example 2 (Fig. 11), was used to extract the data and to match dimensions (by finding which dimension id should be used for each row) with the fact table. The final query that inserted data into the fact table is shown in Example 3 (Fig. 12).

To give better insight into the data, an example of one row in each source table (Table 1) that is then transformed into one row into dimensional and fact tables (Table 2) is shown.

IV. RESULTS AND FUTURE WORK

The main benefit from using the tool was that it was much faster and easier than before, when we were writing all queries manually. The end-user does not need to write SQL queries manually. Instead, a step-by-step guide ensures that the whole process is easy to follow, so unexperienced users can easily use the tool. With our test we have proven that a completely web based tool is functional and that it has some benefits compared to manual ETL. The main benefit is that the tool is intuitive and easier to use than manual ETL. With these benefits, it is possible to use the tool in education while students learn the ETL process. Because no installation is required and more users can use the tool at the same time, this tool is more convenient than traditional ETL tools

However, although we were able to extract data, there were some limitations:

- No aggregate functions were implemented so the profit value could not be calculated and stored into the data warehouse;
- The source database must use primary keys so that the ETL tool can automatically recognize connections between tables. Further on, it is expected that the name of the attribute by which two tables are connected is the same in both tables;
- As a result, we can say that PostgreSQL can quickly process queries shown earlier while MySQL is much slower. The solution could be implemented in a form that MySQL extracts and processes a smaller number of records (for example, 10000 records at once);
- Since data contained some Croatian characters, there was no way to change these characters automatically.

We can say that the tool has limitations and there is definitely space for improvement (we could add aggregate functions, we could speed up the process of extraction and load, parallel extraction from multiple sources is also something that one can consider, etc.).

In the future, detailed tests with more concurrent users are needed. In addition, the idea is to expand and upgrade the performances of this tool to be competitive (for example, in speed) with some traditional tools. In addition, it is planned to completely automate the entire process with ontologies.

V. CONCLUSION

In comparison to traditional ETL tools, the presented ETL tool makes it easier to extract data. It also helps during the ETL process by showing hints of what to do next. But still, the knowledge about own data sources is needed to know what is in which table and user must know what he wants to get at the end.

Since the tool is still a prototype, limitations are present; but, at this point in time, it is possible to create a small data warehouse. Flexible implementation ensures that new features can be easily added.

The main benefit is that it is completely web based, allowing multiple users to use it at the same time. Further on, with integrated step-by-step guidance, even users not familiar with the ETL process can use the tool and learn along the way. Finally, no installation is required.

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

[4] C. Ballard, D. Herreman, D. Schau, R. Bell, E. Kim, and A. Valencic, Data Modeling Techniques for Data Warehousing - IBM redbook, IBM Corporation: International Technical Support Organization, 1998.

[5] M. Novak and K. Rabuzin, “Prototype of a web ETL Tool,” International conference on data warehousing and knowledge discovery, Unpublished

[6] R. K. Vangipuram, V. Sreekanth, and B. Rangaswamy, “Implementation of web-ETL transformation with pre-configured multi-source system connection and transformation mapping statistics report,” International Concerence on Advanced Computer Theory and Engineering (ICACTE’10), IEEE Press, vol. 2, Aug. 2010, pp. 317-322, doi:10.1109/ICACTE.2010.5579100

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