e-System for Automatic Data Migration

Andreea Marin*, Ciprian Dobre*, Decebal Popescu*, Valentin Cristea*
* University POLITEHNICA of Bucharest, Romania
E-mails: {andreea.marin, ciprian.dobre, decebal.popescu, valentin.cristea}@e-caesar.ro

***Abstract*—Interoperability represents an important issue in developing successful e-Services applications. Many times data is obtained from services belonging to one organization and must be used within a service under the domain of another one. Automatic data migration and transfer utilities are not sufficiently addressed. There are various ways of moving data between files and databases, all with the common inconvenience that the structure of the data is fixed. Today's users' requirements are for data transfer tools that provide increased flexibility. We present a solution for automatic data migration that is able to handle data transfers between different database instances. The solution is able to work with dynamic data schemas, and requires minimal user input and interaction. The solution was evaluated as a case study in a real-world e-System designed to automatically collect report data.** *(Abstract)*

Keywords- data migration, automatic data conversion, interoperability, e-System

I. INTRODUCTION

Many e-Systems are made by composing the functions provided by services running in different organizations. Such services often use similar data, stored in different ways. Service composition often uses the data provided by one service as input for other service. In this case interoperability, the way data is semantically linked between these two systems, becomes a critical issue.

Organizations usually collect large volumes of data in their internal databases. In many cases, the stored data refers to the company structure, employees, customers and projects. With the development of new database systems, companies might want to change the database in use and transfer large amounts of data. Moreover changes in company structure and number of employees might generate modifications in the database schema. As a consequence, not only that the underlying database system is changed, but the data transfer would occur between two different schemas.

Data Migration is not an isolated process. Usually implementing a new application which requires data already existing in current storage devices and used for other applications, is a premise for data migration. It is estimated that the industry spends as much as $5 billion on data migration, considering software services and consulting [1].

In this paper we present a system designed to automate the data migration between various data sources. The solution is able to handle data transfers between different database instances. It is designed to work with dynamic data schemas, and requires minimal user input and interaction. The solution was evaluated as a case study in a real-world e-System designed to automatically collect report data.

The paper is organized as follows. In Section 2 we present related work. Section 3 presents the architecture of the data migration e-system. Details regarding the implementation of the system are provided in Section 4 and Section 5 describes a form of usage of the system. Section 6 gives conclusions and presents future work.

II. RELATED WORK

Data migration has been previously studied in research literature. A number of solutions [2, 3, 5] were previously proposed, each having various limitations. A number of research papers ([1]) analyzed the pros and cons of such solutions, presenting clear explanations about their differences and problems.

In [1] the authors make a distinction between data migration and traditional data moving solutions. Furthermore, the authors make an analysis about how large volumes of data can migrate between database systems and what are the steps in developing an efficient migration strategy.

At present there are various methods for performing data import or export. Various software applications as well as database systems offer the possibility for exporting or importing data. Furthermore they allow exporting data in various known formats: csv, xls. Consequently they allow data import from these types of files also.

Previously, several research projects and software products tackled ideas related to our proposed solution. We next present details about several such solutions, which represent the state-of-the-art of this area. According to [2], SQL Server Integrated Services (SSIS) provide the developer the necessary means to develop workflows for data transfer. The workflows take the form of packages that contain data and control flow information, used both for creating mappings between source and destination, but also for connecting them. Moreover, the means for data transfers are provided also by SQL Server Management Studio Express, in the form of tasks that can be executed on top of the database. This utility transfers data in known file types (csv, xls) or in Oracle and SQL Server Databases having the same schema. On the other hand, the transfer depends on the Oracle Database System's version (10g, 9i).

Talend Open Studio is a very complex data integration and business modeling software product. According to [3]

using specific jobs in Talend Open Studio can help transfer data between any two sources. It offers wide support for the most important database systems and for the best known file types. What it may be considered as an inconvenient is that the schemas of the source and destination entities have to be known at design time for the developer to be able to create mappings between them.

Unlike previous work, the solution presented in this paper tries to automate the steps in converting the data and automatic mapping of fields, even if they pertain to different data schemas. The solution can be used to automate many of the steps required when services require the data produced by other services, or the migration of data between different databases.

III. The Architecture of the Automatic Data Migration System

The architecture of the Automatic Data Migration System is presented in Figure 1. The system maps data between two data sources. It can then dynamically transfer the data whenever required.

The system first loads the data schema (fields, relationships, etc) from the first data source. The user is then presented with an intuitive representation of the data schema. The user has full control over the data migration process using this data interface. On the other hand, the system includes mechanisms to automatic recognize data types and map data fields belonging to different data sources. It can be instructed to move data on various triggered actions, even if conversions among data types are required.

The architecture consists of specialized modules that interact with each other maintaining a continuous flow of data. The main components are as follows. The three main modules of the software application are the **rule mapping module**, the **data conversion module** and the **data access layer**. These modules are controlled indirectly by the graphical user interface.

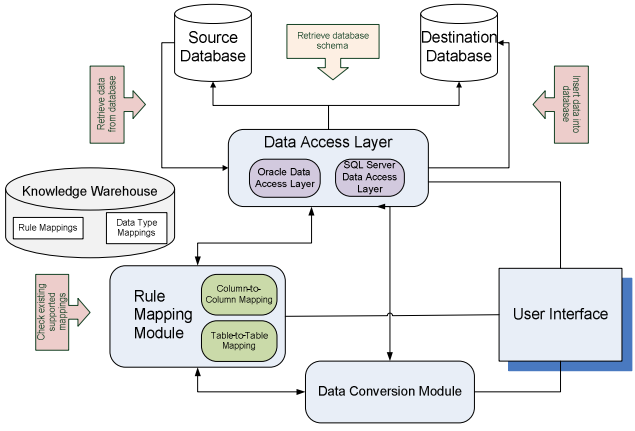


Figure 1. Application Structure.

The data access layer retrieves the database schemas in order to be displayed by the user interface. In addition it is used for database specific operations such as data retrieval and insertion.

The rule mapping module can be used to register mapping rules in the system, to manage and send for execution against the involved databases. The rule mapping module checks the validity of the rules using a repository. The repository consists of several implicit and explicit mappings that are supported by the application.

The data conversion module is used for the data conversion that must occur during data migration. In addition, a knowledge warehouse is used for storing conversion mappings between known data sources.

IV. Implementation Details

The implementation of the e-System supports data migration/transfer between data sources without having prior details about the structure of the data .The implementation of the automatic data migration system considers several of the most known data sources.

In the majority of batch data transfer applications the schemas must be defined at design time. For example, .NET's Typed Data Sets is a data mapping that can be used in design [6]. However, such an approach cannot be used for implementing the migration system. In our system no schema is defined at design time and this increases the usability of our system for non-technical users. This is an important improvement because many data migration projects have to be developed and run by database specialist.

The system allows the user to perform data migration activities on his own, without him having advanced technical knowledge over database systems. Furthermore the application supports migration between similar and different databases. For example, one of the first features supported was data migration between two SQL Server databases. In addition, the second important feature supported was migration between an Oracle database instance and a SQL Server one.

Concerning security issues, the only information the user has to know is the credentials of these databases. As a consequence, both databases must allow remote connections and the user must have administrator rights on the system on which he is running the data migration system.

Software requirements for the proposed system include Microsoft .NET Framework 3.5 and Microsoft SQL Server 2008 Express. Microsoft .NET framework is used for the implementation of all the three modules. The database is the main component of the Knowledge Warehouse, used to store mapping rules information.

In the following paragraph we present details about the implementation of the main components.

A. User Interface

The user interface supports several functions:

- Authentication
- Creation and management of mapping rules
- Display of data schemas, conflicts or resolutions
- Display of migration results

The user inserts the credentials for the source and destination data sources. The user interface supports a wide range of authentication mechanisms, designed to support connections to various databases and flat file sources. After the credentials are verified and a connection is created, the data schemas are retrieved from the both data sources and they are displayed. All information is displayed in a web portal, designed using.NET platform TreeViews. An example of this portal is presented in Figure 2.

Using the interface, the user can browse through the tree structure and select the columns and tables he wants to use for constructing the mapping rules. The mapping rules are dynamically created by selecting a source end point and a destination end point. The end points are selected from a list of available data sources.

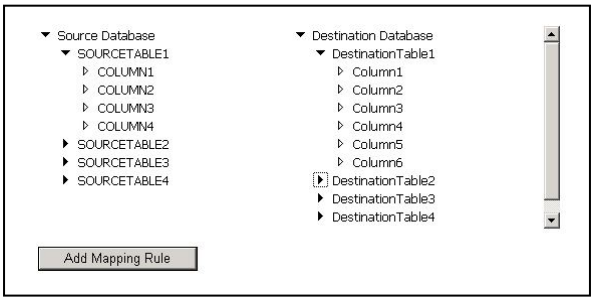


Figure 2. Data Sources Structure.

After the mapping rules are created they are added into a list and can be further reviewed and managed. The list, presented in Figure 3, is available for the user to delete the rules he does not want to continue with. After the rules are saved the user can proceed to the rules checking and execution. If all the mapping rules have structures similar to the supported mapping rules, the migration can occur. If not the user is displayed with a list of conflicts and the possibilities to resolve them. This feature is explained in the following sections.

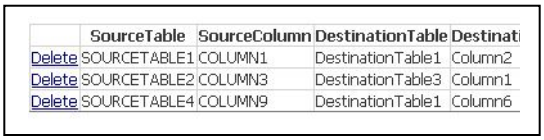
B. Rule Mapping Module

The rule mapping component is used to create mapping rules according to the specifications of the user, to verify the correctness of the rules and, if all the conditions are verified, to further send the rules for execution against the database. The system supports two types of mappings: column-to-column mappings and table-to-table mappings.

The column-to-column mapping is described as follows. By means provided by the interface the user can select the source and destination location of the data. The user can choose any column from the first database (the source column) and any column from the destination database (the destination column). Also, this module uses the Knowledge Warehouse component to automate the process of column mapping. This component includes various predefined rules of mappings columns. For example, if two columns have the same name (for example the name of an employee or salary) then the user is already presented with the mapping rule. Also, if the user previously selected a mapping rule and the same two columns are again used in another selection, again the two columns are already mapped. However, the user has full control over these predefined mappings.

After the two columns are selected the user can use an interface button to register his mapping in the system. In the end, when all the mappings are registered the user instructs the system to send the rules to be processed against the databases.

The table-to-table mapping is similar to the column approach, but instead of columns, the end point entities are tables. This feature was added to support the case study reporting system that is presented in the next Section. This reporting system considers that there is a slight possibility that users have similar data structures in both source and destination databases. When this kind of mapping is performed the system checks that the involved tables have the same structure exactly. This is equivalent with comparing two hash tables having the same key values, in this case, the keys being the column names. If the column names and data types are not the same the migration cannot be performed and the user is notified accordingly, being required to act in concordance with the best suited solution for his needs.



	SourceTable	SourceColumn	DestinationTable	Destinat
Delete	SOURCETABLE1	COLUMN1	DestinationTable1	Column2
Delete	SOURCETABLE2	COLUMN3	DestinationTable3	Column1
Delete	SOURCETABLE4	COLUMN9	DestinationTable1	Column6

Figure 3. Examples of Mapping Rules.

The rule mapping module also allows managing the created rules. Before starting the data conversion, the user can check the created rules and decide whether he wants to use all of them or not. Both, the user interface and the rule mapping module provide the ability to delete the rules considered unsuitable.

The module also supports the management of user generated errors, such as creating an incorrect mapping. An incorrect mapping can be defined as a mapping whose end point entities do not have the same type. For example a user can create a table-to-column mapping, which would generate

serious damage to the data migration process if not discovered prior to the start of rule execution.

C. Data Conversion Module

This module assists the rule mapping and migration functions of the e-System with data conversions. This has to be performed according to the user needs and in such a way that data is not altered during the conversion. An incorrect data conversion can negatively affect the applications relying on that data as well.

The module supports two types of conversions:

- Implicit conversions
- Explicit conversions

Implicit conversions are made between well-known database data types [4]. Examples include conversions between SQL Server, Oracle or other databases, such as Binary to Raw, Image to Long raw and others (see Table 1). Implicit data conversions are automatically created by using predefined rules provided by the Knowledge Warehouse component.

The explicit conversions are used for conversions between non-compatible data types. For example, for some situations the user might want to convert numerical data types into strings or viceversa. Such conversions are part of the requirements that the user is presented with an large collection of data conversions. Table 2 presents several such explicit data conversions.

TABLE I. EXAMPLES OF IMPLICIT DATA CONVERSIONS.

Data type	SQL Server	Oracle	MySQL
boolean	Bit	Byte	N/A
integer	Int	Number	Int Integer
float	Float Real	Number	Float
currency	Money	N/A	N/A
String (fixed)	Char	Char	Char
String (variable)	Varchar Nvarchar	Varchar Varchar2	Varchar
Binary object	Binary Varbinary Image	Long Raw	Blob Text

To prevent possible data losses, when the user requires such a conversion, he is presented with a warning message stating that continuing this action might alter the data. As a consequence, the user must agree on continuing with the data conversion. If the user does not find the data conversion suitable for his requirements he has two choices:

To perform just the data conversions that are considered safe and then alter the destination database; or

To stop the whole migration process, alter the destination database and restart the migration process.

TABLE II. EXAMPLES OF EXPLICIT DATA CONVERSIONS.

Source data type	Destination data type
String	Int
Boolean	Int
Boolean	Bit
Int	String
Boolean	String
Char	Boolean

This component is also capable of automating the conversion processes. For implicit conversions the system is able to recognize formats and correctly manage data migration between columns of compatible data types. Also, the Knowledge Warehouse stores information for both implicit and explicit data conversions. Whenever a data conversion rule that does not exist in the repository is created by a user, and the rule is validated by the data conversion module, it is saved for further usage in a temporary storage space. The system administrator checks the temporary tables periodically and if the rules are in concordance with the system's requirements and purpose, they can be added to the permanent mapping tables.

V. A CASE STUDY

The application presented in this paper was tested and used in an e-Services system for public administration reporting services. The e-Services system is able to provide optimized automatic data transfers, document workflows and business reporting of business organizations. Its main purpose is to optimize businesses improving management of routine tasks such as periodical reporting data or automatically managing interactions between the organization and the public administration.

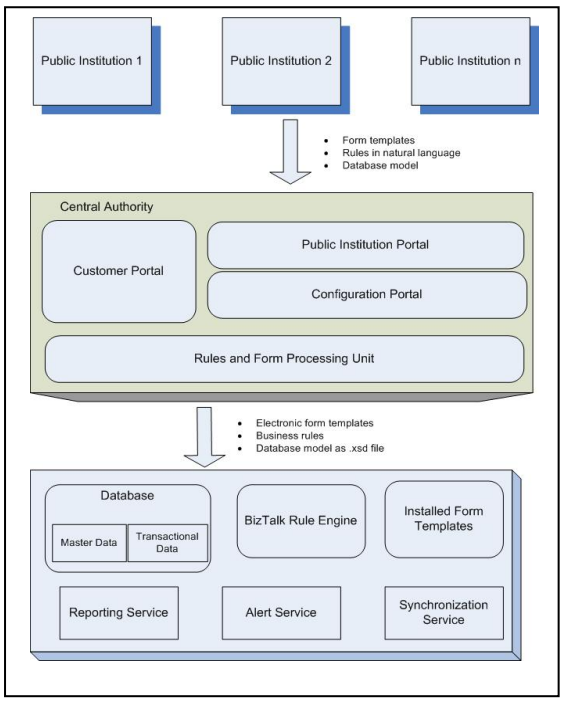


Figure 4. The Architecture of the reporting e-System.

The architecture of the e-System (presented in Fig. 4) involves a high degree of interaction between the system and the public institutions. In consists of several software components, called Processors, responsible with the automatic generation of reports based on the data available within a business organization. A Central Authority is responsible with communicating with all processors, updating their rules and report templates and managing their processes. In addition, public institutions interact with the Central Authority and provide information regarding the reports using a natural language.

The role of the data migration software is to provide the initial data for the reporting system. The reporting system in discussion has its own database system. At the first use of this system, the only information available is the database schema. For the initialization a series of internal data is needed. The data is considered to reside partially or totally in the user's own database. For the transfer of the data between the two databases, the presented software is used. The issues arising from this approach are which data is needed for the transfer, how the data is transferred and what is the structure of the source and destination database schemas.

Based on this architecture, we developed a pilot implementation based on the business realities in Romania. The basis of such an implementation consisted of a series of specific Romanian public administration documents used in domains such as social insurance, environment and fiscal reporting. The analysis of the reporting processes between businesses and public authorities revealed the existence of 4 classes of reports, for reporting fiscal, social, environmental and statistical data.

Such reports can generate a lot of documents and bureaucracy that can have a negative impact on performance of the private business enterprises.

For the pilot implementation we concentrated, in particular, on the 010 Fiscal Registration Declaration, officially known as declaration of amendments for judicial persons, associations, and other entities without judicial personality. The sensitive data that has to be inserted into this report consists of the identification data of the taxpayer and the categories of declaration tax obligations according to the law, hereinafter called fiscal vector. The fiscal vector can be defined as the permanent obligation of the taxpayer and consists of data regarding the following financial aspects: VAT, excises, petrol tax and natural gases from the internal production tax, gambling tax, profit tax, fiscal royalties, micro corporation income tax, wages income tax, special taxes such as: social health insurance tax, unemployment tax, professional illness and accidents tax, social insurance tax.

In this context the migration system optimized the overall performance of the reporting system by ensuring a transparent data retrieval process. It allows the user to easily manage the data mapping processes for filling the required reports.

Without the migration service, the user should be presented with mechanisms to export the data required by the reporting system manually. The data cannot even be directly imported, manually conversions being needed for many of the fields required. Using the migration service there is no need to have a data operator to deal with the data, many of its tasks being performed automatically. Therefore, the system ensures a lower operating time and fewer errors when moving the data.

VI. Conclusions and Future Work

In this paper we presented a system designed to automate the data migration between various data sources. The solution is able to handle data transfers between different database instances. It is designed to work with dynamic data schemas, and requires minimal user input and interaction.

Organizations usually collect large volumes of data in their internal databases. In many cases, the stored data refers to the company structure, employees, customers and projects. With the development of new database systems, companies might want to change the database in use and transfer large amounts of data. Moreover changes in company structure and number of employees might generate modifications in the database schema. As a consequence, not only that the underlying database system is changed, but the data transfer would occur between two different schemas. In this context, the system presented in this paper can optimize the effort needed to correctly manage the migration process. As a case study, for example, we evaluated the solution in a real-world system implementation designed to automatically collect report data. The migration system leaded to the

optimization of the overall performance of the reporting system by ensuring a transparent data retrieval process. It allows the user to easily manage the data mapping processes for filling the required reports.

In the future we plan to further extend the migration system with advanced mechanisms for data type recognition and conversions. We plan to design a solution where the Warehouse repository can be dynamically updated from a central repository of knowledge.

ACKNOWLEDGEMENT

The research presented in this paper was supported by the international project "PrO", developed by Centre for Advanced Studies on Electronic Services (e-CAESAR). The work has been co-funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

REFERENCES

[1] P. Howard, "Data Migration", A White Paper by Bloor Research, October 2007.

[2] M. Otey, D. Otey, "Microsoft SQL Server 2005 developer's guide", Microsoft, 2005.

[3] Talend, "Talend Open Studio, User guide", Accessed on June 29, 2010, From http://www.talend.com/index.php.

[4] J Paul, "SQL Data Types", Accessed on July 1th, 2010, from http://onlamp.com/pub/a/onlamp/2001/09/13/aboutSQL.html.

[5] J. Morris, "Practical Data Migration", British Computer Society, 2008.

[6] Microsoft, "Implementing Data Transfer Object in .NET with a Typed DataSet", Accessed on July 4th, 2010, from http://msdn.microsoft.com/en-us/library/ff650461.aspx.