International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34428-34432

Toward a proposal of alternative development for data migration using dynamic queries generation

Johan A. ROMERO-RAMÍREZ
jhrom@hotmail.com
Engineering Faculty, Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
Carlos E. MONTENEGRO-MARIN
cemontenegrom@udistrital.edu.co
Engineering Faculty, Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
Paulo A. GAONA-GARCIA
pagaonag@udistrital.edu.co
Engineering Faculty, Universidad Distrital Francisco José de Caldas
Bogotá, Colombia

Abstract- This article presents an ETL (Extract, Transform, Load) prototype called Valery as alternative approach to migration process which includes a compiler for dynamic generation of SQL queries. Its main features involve: SQL dynamic generation, a set of configuration commands and an environment for file uploading. The tests using the Northwind academic database and an individual environment. The model implementation uses flat files and SQL as query language. Finally, there is an analysis of the results obtained.

Keywords: Schema Mapping, Data Transformation, Data Migration, ETL

1. INTRODUCTION

John Morris (2001) [1] defines the data migration as the selection, preparation, extraction, transformation and movement of the suitable data with the right quality, to the the right place and with the dismantling of the legacy data sources.

Some of these projects may have the scenario where the information sources involves flat files [7] and the target system is a relational database and SQL-based [9].

This specific domain presents the problem that usually structures and formats of the legacy data differ or are incompatible with the relational design of the target system and in some cases unexpected changes are raised in these structures.

Therefore, the process must resolve two common and complex tasks in the migration process [8] ((: i) Associate the different schemes mapping related elements between the source and the destination (*mapping*), ii) react effectively to the specification schema changes (*schema changes*).

The Figure 1 shows an example of different sources. The scheme of the legacy system presents a different format from the target system related tables and the arrows represent the correspondence between the source-target schemes fields. It's possible to see the differences between the corresponding formats.

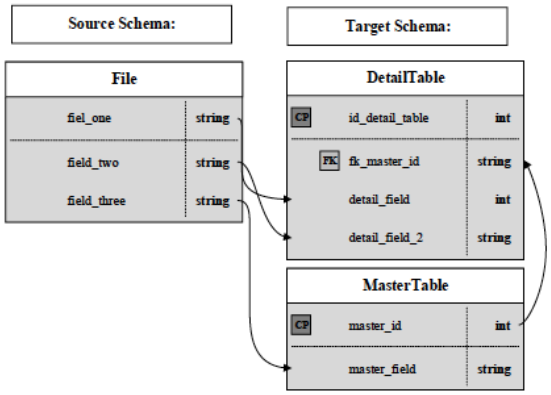

Fig 1. Schema representation with a non-linear connection.

This paper presents the features of a solution called *Valery* as alternative approach to data migration for heterogeneous data sources where the source implies flat files. This is based on the hypothesis that it is possible to create a model that exploit the advantages of standardization and functionality existing by SQL language (Structured Query Language) [10] and the RDMB (Relational Database Management System) [11] onto a model that addresses the problem of mapping exposed and delegates the factors of performance to the infrastructure hardware and legacy data size.

2. FEATURES

Valery approach has the following characteristics:

Conversion of the flat file into a temporary table in the system target: Flat files are converted to a temporary table in the database manager. The advantage of this approach enables subsequent handling by the SQL statements in a transparent manner between the schema source and destination.

Specification of generic cases for the mapping between the schema source and destination: Mapping patterns are

identified to encapsulate the correspondences between the data sources and target tables. This allows adapting the model to a wide range of migration and integration cases which comply with the generic features and assure independence in the relational structure.

Specification of *SQL* statement associated with each case of migration: Different SQL queries needed to solve each generic case for migration and integration are identified.

Generation of dynamic queries: Represents the components associated with the generation of dynamic *SQL* queries. Query Builder creates dynamically the sentences due to a generic pattern recognition and a consistent configuration of the destination relational model. For this, *Valery* has a command console that receives the parameters associated with a specific migration project.

Evaluation with specific data: Implementation was carried out using the *NorthWind* model and an own model. The results indicates a satisfactory performance in quality and runtime.

3. VALERY MIGRATION PROCESS

Mapping specification: The first step is to configure the source with the relational database format. This is an interface that allows to capture of such a configuration in an explicit defined commands.

Load source files: In this step, loading flat files for validation and a migration of the flat file in a new temporary table within the database target system is made for subsequent use in the query execution.

Creation of queries: The system generates the specific SQL queries according to the settings made.

Execution of SQL queries: Running SQL queries created and displays the report of results thrown by the database manager.

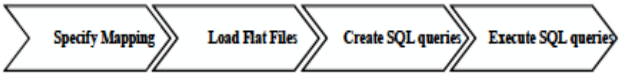

Figure 2. Valery Process.

4. STATE OF THE ART

Different jobs have been for the automatic execution of mapping schemas. To determine the correlation of patterns this works propose techniques that exploit different types of information, such as definitions, data types, structures of schemas and instances of data [2].

As reference works in the development of some components of the proposed model we are presented *QuickMig* tools [2] and *HumMer* (Humboldt Merger) [3]. *QuickMig* is new approach semi - automatic for the identification of semantic correspondences between the sources of information schema elements. *QuickMig* presents a series of new techniques that exploit example cases, domain ontologies and the re-use of existing allocations to detect not only the correlation of elements but also their expressions of mapping. *QuickMig* includes new mechanisms to effectively incorporate the knowledge of the domain's users in the comparison process. The results give a full evaluation using diagrams and actual data [2]. On the other hand *Hummer* is a tool *ad-hoc*, that allow the fusion automatic of data using extensions SQL for heterogeneous schemes with data duplicate or in conflict.

5. OVERVIEW

This section provides a review of *Valery* approach to the correlation of schemes.

a) Considerations

Valery implementation requires knowledge of the domain of the migration process. As also outlined in [2] migration processes need knowledge about the origin and the target system. Unfortunately, such knowledge may not be always available in one place. Knowledge about the target system may be available in the provider, while only customers can provide a detailed origin systems knowledge.

b) Scope

The solution is applicable to projects that could get information bequeathed as flat files and where there is a SQL-based database server (RDMB).

Valery offers a generic case of migration domain. These cases include data with foreign keys, different types of data, updating data processes, independence from flat file formats and independence of the database destination complex.

c) Cases

Linear correspondence of the flat file and database structures that do not include relational data. In Figure 3 is possible to see an example of considered cases. The associated fields have a linear relationship with the destination schema and show differences in nomenclature and data type.

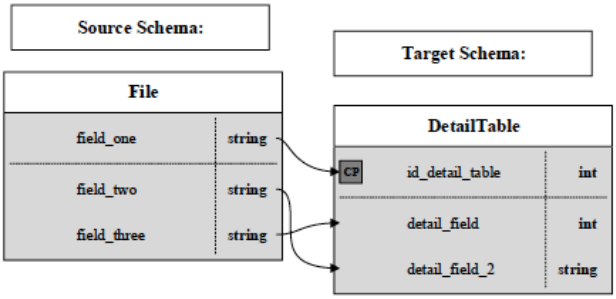

Figure 3. Schemas representation of with a linear correspondence.

There is a linear correlation between the flat file and the structures of the database due to referential integrity presenting tables: the original file must populate all tables in a single transaction (see Fig. 1). Data is generated massively associating foreign keys dynamically. In addition, there are differences in nomenclature and data type.

d) Algorithmic theoretical foundation

Valery provides a set of rules for the expression of associated semantic generation of SQL queries and where should be *correlated* data from the data source and the database file destination. The system makes reading of the commands and

generates the corresponding sentences type *INSERT-SELECT-UPDATE* by a generator queries (*Query Engine*). Script specification depends on the complexity of the destination database and of the different levels of the data base relational hierarchy.

The methodology in the generation of SQL queries collect (see Fig. 4): (i) recognition of the SQL statements associated with each case, (ii) decomposition grammar of each sentence, (iii) definition of parameters, (iv) valid from the reading of the sentences *tokens* construction target, (v) creation of the keywords of the compiler according to the parameters, (vi) identification of syntactic trees. This allows one theoretical basis for the modeling and design of the query engine that will take as input the configuration commands according to a specific migration project.

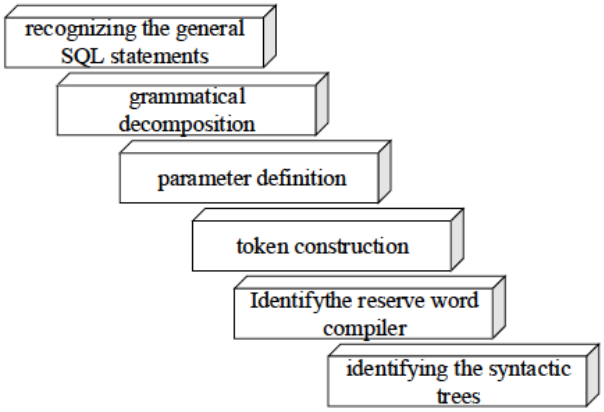

Figure 4. Valery methodology

As an example, the grammatical breakdown of target SQL query associated with case (1) in Figure 5 is displayed.

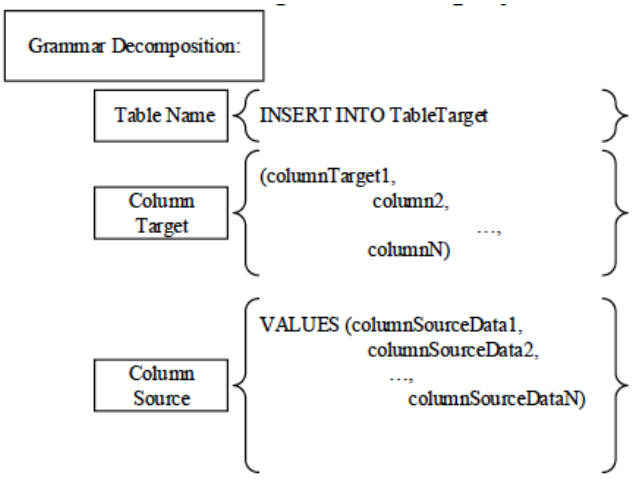

Figure 5. Grammatical case of inclusion one decomposition.

The model breaks down the previous sentence in the following grammatical fragments that symbolize the commands in the system for their respective configuration:
(a) Table name.
(b) Set of fields in the table destination.
(c) Set of fields in the table origin.

A graph (Fig. 6) representation and representation in first order algebra allow to obtain the following relevant attributes for the *Query Builder* design.

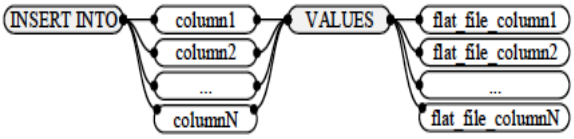

Figure 6. Graph of a SQL statement.

The graph shows the following relevant properties for dynamic generation algorithm:
(a) The root node represents the keyword INSERT INTO.
(b) The next leaf node presents the schema columns.
(c) Equivalent to the keyword VALUES inner node is equal to the number of columns of the template.
(d) The length of the path from the root node to the leaf node is proportional to the number of columns.
On the other hand, a representation in first order algebra of can be represented to a mathematical formalization of the objective statement:

TargetTable ← *TargetTable* ∪ *_{(value1, Value2,..., ValueN)}*

Due to the previous theoretical foundation, it is possible to represent the attributes required in script type attribute-value for the correct generation of case one target SQL statements. Below is show the commands required by the *QE*:

Attribute (Command_TT): value (name of the destination table).
Attribute (Command_FS): value (field in the source table).
Attribute (Command_FT): value (field of the destination table).

In addition, the *QE* algorithmic design implements a data structure of the syntactic tree such as a perfect binary tree where the hierarchy of commands depends on the foreign keys of tables. Fig. 7 represents the syntactic tree for a specific case where the representation of commands is modeled.

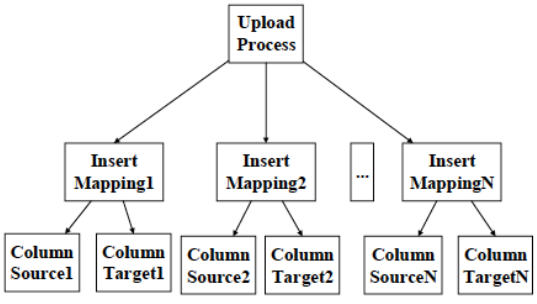

Figure 7. Binary tree used by the Query Builder.

It is noted that the route of the syntactic tree does not require rearrangement or searches to meet a predefined SQL syntax [4], is only requirement the algorithm

implemented functionality of insertion into the binary tree elements.

This tree insertion in the binary tree has an O (log n) complexity [5] and is recognized as a problem class P (treatable) [6].

6. VALERY ARCHITECTURE

Valery components include: (i) **use of flat files:** flat files as legacy information due to the support that lies in the integration of various applications in these formats. (ii) **GUI:** this layer deploys all components related to the design, data capture, the configured schema validation, loading flat files, reports of the processes and operations carried out by the users to interact with the system. (iii) **Query Engine**: represents the components associated with the dynamic generation of SQL statements. To do this follow the exposed methodology (Section 5.4.). The *QE* is responsible for compiling the predefined rules and generate SQL statements according to the request made by the user. This client/server architecture allows *scalability* and *concurrence* of multiple migration processes. (iv) **Data access layer**: are the components dedicated to establish the connection with the RDBMs, affect the target database according to each charging process and the results of the application.

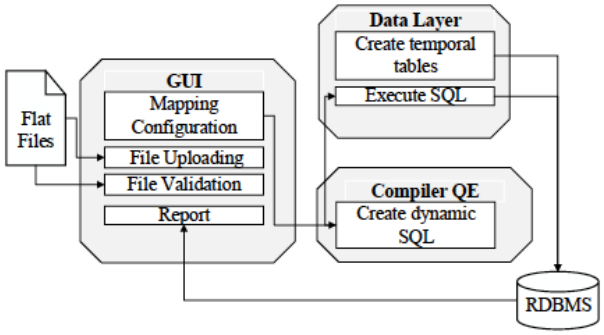

Figure 8. Valery architecture model.

Valery follows the architecture presented in Figure 8. The core (*QE*) is based on the grammatical breakdown of SQL statements that satisfy the cases of migration and integration. A set of commands and rules where the QE works as an interpreter of these commands is defined in the GUI.

7. TESTING SCENARIO

Test database (*Northwind* and the own model) was used as target source, SQL Server as RDBMS and several flat files (*.csv) structurally separated with table target. With this testing scenario created verified the following accounts:

I. Transformations of data complexity: For this purpose a set of migration process affecting different tables using as input files that have structurally independence with the target database is defined.
II. Loading the flat file as an additional table in the database destination: this leads to a better performance of the SQL queries execution.
III. Tracking and audit: the statements to be executed in the respective RDBMS uses all the functionality offer by the RDBMS for tracking and audit.

Technical hardware and software for testing: Intel ® Core ™ 2 Duo Processor, Windows 7 Home Premium (64-bit), 4 GB DDR2 800 MHz memory, 500GB HDD (5400 RPM, Serial ATA), Sql Server 2008 ®.

Migration *Northwind* processes involved 32 SQL queries generation while that for the own database processes involved 16 SQL queries generation. The data types of the formats included strings, dates and numeric values.

```
INSERT INTO Categories
(CategoryName,Description,Picture)
SELECT cmpExcel5,cmpExcel8,cmpExcel11
FROM TablaExternaV10
```

Fig 9. Example 1 . INSERT queries generated for the Northwind database.

```
INSERT INTO Products
(SupplierID,ProductName,QuantityPerUnit,Un
itPrice,UnitsInStock,UnitsOnOrder,ReorderLe
vel,Discontinued)
SELECT
(SELECT TOP 1 tabla0.SupplierID FROM Suppliers AS
tabla0
WHERE tabla0.ContactName = cmpExcel25),
cmpExcel28,cmpExcel31,cmpExcel34,cmpExc
el37,cmpExcel40,cmpExcel43,cmpExcel46
FROM TablaExternaV10
```

Fig 10. Example 2. INSERT queries generated for the Northwind database

```
UPDATE Orders
SET Freight = campoExcel81,
ShipName = campoExcel84
FROM TablaExternaV10
WHERE OrderID =
(SELECT TOP 1 tabla0.OrderID
FROM Orders AS tabla0,
Customers AS tabla1,
Employees AS tabla2
WHERE tabla0.ShipVia = cmbDinamico51
AND tabla1.CustomerID = tabla0.CustomerID AND
tabla1.ContactName = cmpExcel59
AND tabla2.EmployeeID = tabla0.EmployeeID AND
tabla2.FirstName = campoExcel66)
```

Fig 11. Example 2. UPDATE queries generated for the Northwind database

Tables 1 and 2 show the results of the tests according to the model of testing used.

Table 1 Evaluation of results of the query execution.

BD	Querys	Time (sg)
NorthWind	32	0.54
SAP	16	0.26

Table 2 Average response of processes according to file size.

Size	Average: Upload and Execution
1mb	12sg
10mb	249sg
100mb	3526sg
1000mb	17513sg

In table 1, it is possible to observe performance in the generation of SQL queries. It obtains the expected result of the syntax (see fig. 9, 10, 11) in the generation of queries that will be subsequently executed in the RDBMS.

On the other hand in table 2, is possible to see that for scenarios where the files have a weight of larger system presents an unfavorable performance. This is due to the conversion of the flat file into the temporary table is proportional to the size of the file and the query execution are also proportional to size the temporary table.

8. CONCLUSIONS

The approach presented by *Valery* was experimentally evaluated using data in a test environment. It is quite effective in creating well-formed SQL queries according to the defined specific domain. Moreover proportionally migration performance depends of the size file and the database server hardware. Therefore, performance is an independent factor to the generation of SQL queries.

Our approach using generic cases allowed a coverage of different sorts of migration requirements in two testing databases using insertion and update.

Mapping and data transformation conditions could solve for specific tests. The approach allowed encapsulate the problem of mapping settings and delegate the performance issue to the hardware infrastructure and file sizes corroborating the initial hypothesis.

9. REFERENCES

[1] Practical Data Migration, John Morris. BCS. 2012, pp. 7, 8, 9, 10, 77, 164.
[2] QuickMig. Automatic Schema Matching for Data Migration Projects. Christian Drumm. Universität Leipzig, Leipzig, Germany, 2007.
[3] Automatic Data Fusion with HumMer. Alexander Bilke, Jens Bleiholder, Christoph Böhm, Karsten Draba, Felix Naumann, Melanie Weis. Technische Universität Berlin, Germany. Humboldt-Universität zu Berlin, Germany. 2005.
[4] What Is Sql?: Fundamentals of Sql, T-Sql, Pl/Sql and Datawarehousing, Victor Ebai 2012, pp XI.
[5] Mastering Algorithms with C (Google eBook). Kyle Loudon. O Reilly Media, Inc. ". 2009, pp. 206
[6] Metaheuristics. Abraham Duarte Munoz. Editorial Dykinson. 2007, pp. 12.
[7] Implementing a Data Warehouse: A Methodology that Worked. Bruce Russell Ullrey. AuthorHouse, 2007, pp. 93-94.
[8] Schema Matching and Mapping. Zohra Bellahsene, Angela Bonifati, Erhard Rahm. Springer Science & Business Media. 2011, pp. 152, 153.
[9] Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments: Challenges in Service Oriented Architecture and Cloud Computing Environments. Ionita, Anca Daniela. 2012, pp. 210.
[10] Database Management System. Seema Kedar Technical Publications. 2009. pp (4-2).
[11] Fundamentals of Relational Database Management Systems. S. Sumathi, S. Esakkirajan. Springer.