Query-based Data Warehousing Tool

Rami Rifaieh
TESSI Informatique
92 bis Rue Bergson
42000 St-Etienne, France
(33) 477.79.56.03
rrifaieh@tess2i.fr

Nabila Aïcha Benharkat
LISI - INSA de Lyon
7 Avenue Jean Capelle
69621 Villeurbanne, France
(33) 478.93.38.68
nabila.benharkat@if.insa-lyon.fr

ABSTRACT
Data warehousing is an essential element of decision support. It aims at enabling the knowledge user to make better and faster daily business decisions. In order to supply a decisional database, meta-data is needed to enable the communication between various function areas of the warehouse and an ETL tool (Extraction, Transformation, and Load) is needed to define the warehousing process. The developers use a mapping guideline to specify the ETL tool with the mapping expression of each attribute. In this paper, we will define a model covering different types of mapping expressions. We will use this model to create an active ETL tool. In our approach, we use queries to achieve the warehousing process. SQL queries will be used to represent the mapping between the source and the target data. Thus, we allow DBMS to play an expanded role as a data transformation engine as well as a data store. This approach enables a complete interaction between mapping meta-data and the warehousing tool. In addition, this paper investigates the efficiency for a Query-based data warehousing tool. It describes a query generator for reusable and more efficient data warehouse (DW) processing. Besides exposing the advantages of this approach, this paper shows a case study based on real scale commercial data to verify our tool features.

Categories and Subject Descriptors
D.2.11 [Software]: Software Architectures – Data abstraction.

General Terms
Management, Theory, Design, Experimentation.

Keywords
Data warehouse, Meta-data, Query based ETL tools, Mapping expression.

1. INTRODUCTION
Data warehouse system is a collection of data used for decision-making. The success of data warehouses implementation for business intelligence activities implies an increasing demand for new concepts and solutions [1]. It includes growth across platforms, tools, and applications. Meta-data is the essential information that defines the what, where, how, and why about the used data. It can range from the conceptual overview of the real world to detailed physical specifications for the particular database management system [10].

Many applications need to establish a schema mapping from many sources to a target. These applications use a mapping meta-data or mapping guideline to achieve this process. For data warehouse system, ETL tools are used to supply DW with clean data. This process includes a mapping transformation for the data. The maintenance of these tools to provide evolution is a hard task and time consuming. In order to simplify this task, a full implementation of mapping guideline or mapping meta-data is needed.

Therefore, we suggest to generate automatically queries from meta-data to keep the warehousing tool updated. The purpose of this paper is to show how we can supply an ETL tool by queries processing to perform the transformations and load process. For this, we allow DBMS to play an expanded role as a data transformation engine as well as a data store. Therefore, SQL queries could be used in all the construction process, especially in data extraction, cleansing procedures, direct storage, and front-end information delivery. As we also deal with the implementation of mapping meta-data, the automatic generation of queries is an appropriate solution to take advantage of this implementation. Moreover, the automatic generation of queries transformation could be a solution for the personalization process.

We will focus on a case study concerning the architecture of DW and the meta-data to show where and how our approach can reduce process complexity. In this case study, we try to show how it could be easier to acquire and transform data by using SQL queries. We also discuss the automatic generation of these queries from mapping meta-data.

The paper is organized as follows. In the next section we will glance over DW components and its architecture. Section 3 will define the background of this paper including mapping guideline, mapping expression, and their models. Section 4 will describe the design of QELT (Query based Extraction, Load, and Transformation tool). The case study will be detailed in section 5. We will conclude our work in section 6 with the future perspectives.

2. THE DATA WAREHOUSE SYSTEM
Data warehousing is an essential element of decision support, which has increasingly become a focus of the database industry. It constitutes the background to enable business intelligence

DOLAP'02, November 8, 2002, McLean, Virginia, USA.

solution, which lets organizations access, analyze, and share information internally with employees and externally with customers, suppliers, and partners.

2.1 Data Warehouse Components

The construction of a DW is divided to two stages known as back room and front room. The first ensures the building up of the warehouse database. The second provides the restitution of data from data marts in order to fulfil analyst's demands.
According to standard data warehouse architecture, the data warehouses systems are composed of:

- ETL or warehousing tools: a set of tools which are responsible of preparing the data constituting the warehouse database;
- Restitution tools: the diverse tools, which help the analysts to make their business decisions, and;
- Meta-data: brings together the data about all the components inside the DW.

2.1.1 ETL tools

ETL (Extraction, Transformation, and Load) represents the warehousing tools or population tools. These tools are globally responsible for the constitution process of the data warehouse [5]. They must so take into account the sources and targets heterogeneity. Therefore, they should verify the integrity of data from these sources, apply data cleaning, check redundancy, etc.

Moreover, warehousing tools have another challenge to provide maintenance capability, availability, task management, and evolution support. Data integration and reuse possibilities are wide open but not yet very well realized. Although some tools provide reused functions, these solutions are still limited. Indeed, existing functions don't allow users to utilize an existing transformation plan and specify it with parameters to create a new data warehouse.

Otherwise, the existing tools have common limits such as:

- The use of a specific format: most of the tools use an owner format to establish the warehousing process. Thus, the communication between different tools becomes hard. This communication is needed to merge different tools in order to establish the warehousing process.
- The maintenance: the example showed above, proves the complexity of maintenance because a set of programs has to be updated. An update takes place in two steps; one time by updating data in the meta-data, and another time by modifying warehousing programs.
- The limited interaction with meta-data: in this context, meta-data is very passive because it is query limited.

Many commercial products and services are now available, and all of the principal database management system vendors now have offerings in these areas. Thus, a diverse range of tools exists with different capabilities such as *DTS, Data Stage, Sagent, Informatica, Data Junction, Oracle Data Warehouse Builder, DB2 Warehouse Manager, etc*. These tools differ by their performance, data source integration capability, use complexity, platforms, etc. A subset of their capabilities is described in Table1, after being tested with the same type of files.

3. MAPPING GUIDLINE

In this paper, we are interested in the representation of the mapping guideline. By mapping guideline we mean the set of information defined by the developers in order to achieve the mapping between the attributes of two schemas. We identify mapping expression of an attribute as the information needed to recognize how a target attribute is created from the sources attributes.

Table 1. ETL Tools Capabilities

Product	DTS	ORACLE DW BUILDER	DATA STAGE	SAGENT
Performance	Average High with SQL Sever	High With Oracle	Depends on data source	Depends on data source
Maintenance	Updating ActiveX programs	Compiling modified programs	Modifying used objects	Modifying plan content
Reusability	Defined functions	Defined functions	Set of objects or plan	Reusable plan
Access to meta-data	None	None	Meta-data stage	None
Plan for process extraction	Plan with timer	Plan with triggers	Event management	Sagent automation
Data sources (Data integration)	Txt, Via ODBC OLEDB (*MSAcces…*), …	Flat files, Oracle, Sybase, Via ODBC OLEDB (*MSAcces…*), …	SQL Server, Oracle, DB2, Txt, VSAM, Informix, …	SQL Server, Oracle, DB2, Txt, VSAM, Informix, …

3.1 Existing Solutions

Actually, different kinds of mapping guidelines are used for many applications. Traditionally, these guidelines are defined manually during the implementation of the system. In the best case, they are saved as paper documents. These guidelines are used as references each time we need to understand how an attribute of a target schema has been generated from the sources attributes.

This solution is very weak in the maintenance and evolution of the system. To keep updating these guidelines is a very hard task, especially with different versions of guidelines. To update the mapping of an attribute in the system, one should include an update for the paper document guideline as well. Thus, it is extremely difficult to maintain such tasks especially with simultaneous updates by different users.

3.2 Using Of Mapping Expression

We can identify some of the applications where mapping expressions are used.

- Schema mapping: has been studied in [9], [15] and others. For database schema mapping, the mapping expression is needed to define the correspondence between matched elements.

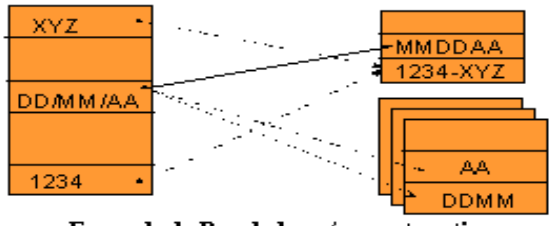
- Data warehousing tool (ETL): includes a transformation process where the correspondence between the sources data and the target DW data is defined. These mapping expressions could be defined in the meta-data of the system [14].
- EDI message mapping: the need of a complex message translation is required for EDI, where data must be transformed from one EDI message format into another [13]. This process includes a mapping between the content of the messages.
- EAI (Enterprise Application Integration): the integration of information systems and applications needs a middleware to manage this process [12]. It includes management rules of an enterprise's applications, data spread rules for concerned applications, and data conversion rules. Indeed, data conversion rules define the mapping expression of integrated data.
- Proprietary ERP (Enterprise Resource Planning): Mapping expression is used for the personalisation purpose of ERP applications [2], [12].

Therefore, we need to put the mapping guideline inside the system meta-data and use this meta-data to reduce the complexity of manual definition of mapping expression.

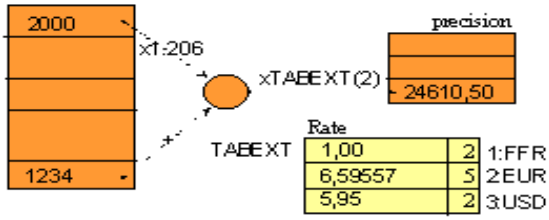
3.3 Mapping Expression Examples

In this section, we will present some examples of the mapping expression identified from different type of applications.

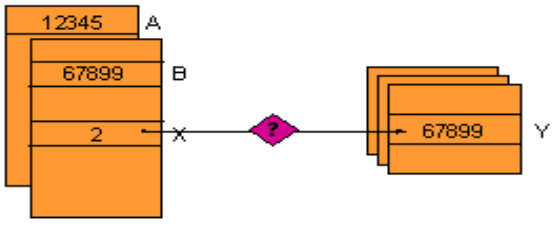
- Break-down/concatenation: in this example the value of a field is established by breaking down the value of a source and by concatenating it with another value (Example.1).
- Arithmetic operation with multiple data: in this case an arithmetic function is defined. This function calculates the result value of the target attribute. The example uses a reference table (rate) in order to calculate the precision (Example.2).
- Conditional mapping: sometimes the value of a target attribute depends of the value of another attribute. In the example, if X=1 then Y=A else Y=B (Example.3).
- Mapping with breaking key: the representation of a row depends on the value of a field. In our example if X=1 this rows represents Enr1 and if X=2 this rows represents Enr2 (Example.4).



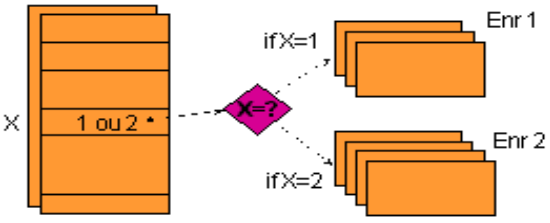
Example.1: Break down/ concatenation



Example.2: Arithmetic operation



Example.3: Conditional mapping



Example.4: Mapping with breaking key

3.4 Mapping Expression Model:

In this section, we will try to define a formal model to represent mapping expression and the mapping guideline. First of all, we adapt the definition of the mapping guideline as a set of mapping expression between two different schema representations.
The used notations are:

- Let $S_1, S_2, \ldots, S_n$ represent n source of data (schema)
- Let $a_1^{S_1}, a_2^{S_1}, \ldots, a_m^{S_1}$ represent m attribute of the source S_1
- Let T represent the Target data container (or target schema)
- Let A represent an attribute of the target T.

Each attribute of the source is associated meta-data. These meta-data are two parts: the attribute identity meta-data and the mapping meta-data. The attribute identity meta-data exists within the description of the source schema. This part of meta-data covers the attribute name, the relation name, the schema name, the data base name, the owner of the data, the domain name, etc. The attribute identity meta-data includes information such as max-value, min-value, range, data-types, etc. An uniform model to represent the attribute identity meta-data and other type of meta-data is described in [7], [6].

The mapping meta-data includes the information needed to achieve the attribute mapping from different sources. This meta-data doesn't exist for every attribute in any schema. We identify this kind of meta-data for non-original schema, it means generated from other existing schemas but having different structure. We denotes as target T the non-original schema that can be generated from different schema called sources.

We admit the representation of attribute meta-data described in [8]. Thus, for each attribute is associated its set of meta-data

including all the different kind of meta-data. $\mu(A)$ denotes the meta-data associated with A. Formally, $\mu(A)$ is a tuple ($\mu_1(A)$, $\mu_2(A)$, …, $\mu_z(A)$) of values. For convenience, the authors in [8] give these values $\mu_1, \mu_2, \ldots, \mu_z$ names. For example, the attribute name is denoted *attrname(A)* and the relation name is denoted *relname(A)*. Therefore, attrname*(A)="Address", relname(A)="Orders", and attrdatatype(A)="String"* represent a set of meta-data values of the attribute A.

Based on this representation, we assume $\mu_i(A)$ denotes the mapping meta-data of the attribute A. $\mu_i(A)$ describes how to generate the target attribute A from other sources. Unlike attribute identity meta-data, $\mu_i(A)$ is a tuple of mapping expression ($\alpha_1(A)$,$\alpha_2(A)$, …., $\alpha_s(A)$). Indeed, $\alpha i(A)$ is a mapping expression applied on multi-sources to generate a part of the target attribute A or to generate a sub-attribute of the attribute A. Therefore, mapping guideline M_G is the set of mapping meta-data μ_i ( ) for all the attributes of the target T from different sources $S_1, S_2, \ldots, S_n$.

Thus, M_G ($S_1, S_2, \ldots, S_n, T$) = { $\mu_i(A_1)$, $\mu_i(A_2)$, … , $\mu_i(A_w)$} where $A_1, A_2, \ldots, A_w$ are the attributes of the target T.
In the Example.5, the attribute A of the target T *has* attrname*(A)="Address",* it is composed of *Street* from the source S_1 and the *ZIP Code* of the source S_s. $\alpha_1(A)$ *and* $\alpha_s(A)$ are the mapping expressions for the attribute A.

Below we will discus these mapping expressions,
lets $\alpha_i = < f_i, l_i, c_i >$, where f_i is a mapping function, it could be an arithmetic function or any other string function such as substring, etc.
Hence, f_i can be applied on a set of attribute, these attributes could belong to one or different sources.

Attribute (f_i) ={ $a_r^{Sr}, a_p^{Sp}, \ldots, a_e^{Se}$ } where $a_r \in S_r$, a_r is an attribute of S_r.
Let $S = S_1 \cup S_2 \cup \ldots \cup S_n$
Thus $f_i : SxSx\ldots xS \rightarrow T$
$(a_r, a_p, \ldots, a_e) \mapsto A$ (or a sub-attribute of A)

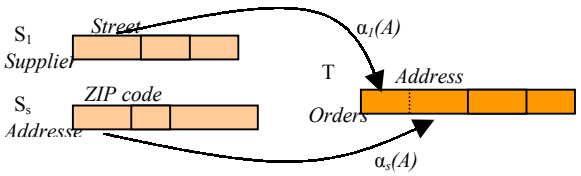
Moreover, l_i is a set of filters for sources rows, we could find a filter $l_i(a_r)$ for each source attribute a_r. This filter can be based on the attribute value a_r itself or based on any other attribute value of the same row. Sometimes, the filters include a joining between attributes of the same source. The use of foreign-key is useful to materialize these filters. In addition, *ci* is a condition on the mapped value of the attribute A. This selection enables us to ignore mapped rows that do not satisfy the condition c_i.
In the Example.5, $\alpha_1(A)$ and $\alpha_s(A)$ are simple kinds of mapping expressions where f_i is the identity function on one attribute without any particular filter l_i or condition c_i.

The Example.6 treats a more complicated situation. We have, attrname*(A)="Value",*
$\alpha_1(A) = <f_1, l_1, c_1>$ and *Attribute*(f_1)={ a_1, a_2 } *where* a_1= *substring(Key_Element, 1,2)* of the source S_1 *and* a_2= *substring(Conversion, 1,1)* of the source S_2 . The function f_1 is the simple arithmetic multiplication, $f_1(a_1, a_2) = a_1 * a_2$.
We have to select the rows of the relation *Element* of S_1 where substring(*Key_Element*, 4,2)="*AB*". This implies $l_1(a_1)$: [(substring(*Key_Element*, 4,2)="*AB*") = True]. We have to select the rows of the relation *Table_con* of S_2 where *Type="£"*. This implies $l_2(a_2)$: [(*Type*="£")=True].
Therefore, we have to select the results of the multiplication to be lower than 45.

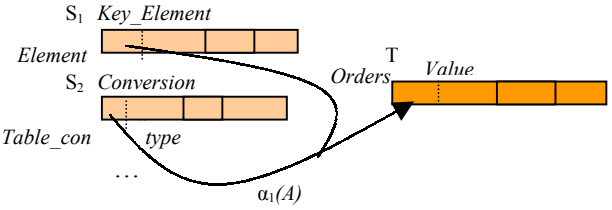
This implies the condition, c_1: *Value* $\leq$ 45.
This model is global for all type of mapping expression between an attribute A of a target schema T and its sources. Generally, in order to generate the complete value of an attribute *A,* we have to concatenate the result of different mapping expression of sub-attributes.

$$A = \text{Concat}\left(\ldots, \underbrace{[f_i(a_r^{Sr}, a_p^{Sp}, \ldots, a_e^{Se}), l_i, c_i]}_{\alpha_i(A)}, \ldots\right)$$

We tried to implement the mapping guideline of type $M_G(S,T)$ (from one source S to a target T) and its set of mapping meta-data including the mapping expression formalism. We applied this formalism to create an automatic ETL tool using SQL queries. This tool is able to read the mapping guideline of the application to generate a complete set of SQL queries needed to extract, transform, and load data inside the DW database.



Example.5: Simple Mapping



Example.6: Mapping with condition

3.5 Related Work:

Cupid [9] is an algorithm of schema mapping between two different representations. It treats the problematic of mapping discovery, which returns mapping elements that identify related element of two schemas. It associates to each attribute or elements of the schema the associated element of the other schema. Cupid does not investigate how to apply the mapping between these elements. Our approach can be complementary for the work of schema matching and it is indispensable to map source and target schema. Indeed, without the mapping expression we can just identify the correspondent elements but we can't apply the mapping to transfer the data.

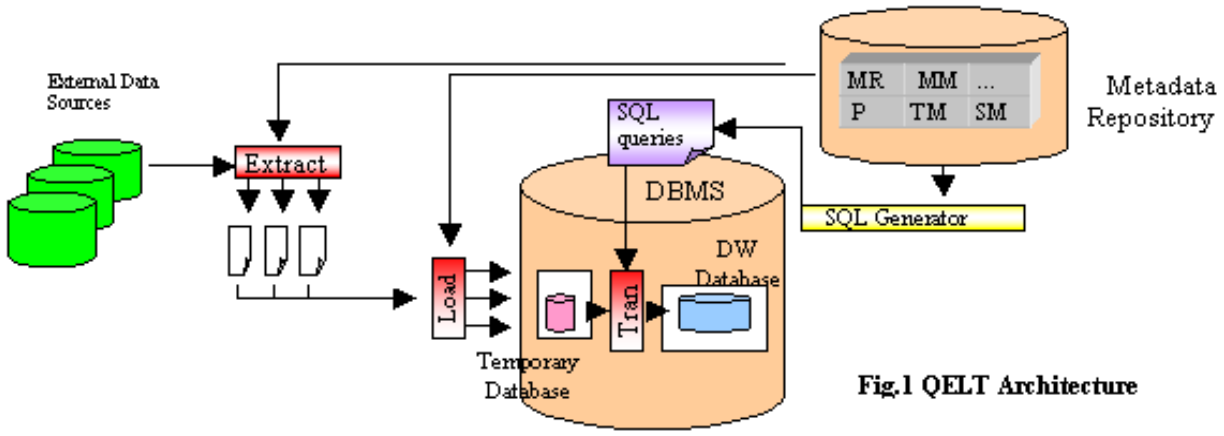


Fig.1 QELT Architecture

4. QELT DESIGN:

In this section, we present QELT, an SQL Queries based Extraction, Load and Transformation tool. The basic idea of this tool consists of using the SQL to create new generation of data warehouse, where DBMS is used as an engine to perform the warehousing process. Therefore, DBMS will be used for two intentions as data storage system and as transformation engine.

4.1 QELT And Existing Tools:

Some commercial tools now support data extraction from different data sources and multiple DBMS to feed the warehouse. Nevertheless, the designer must define the logical mapping between the source and the target schemas and the warehouse schema. Furthermore, these tools do not interact with the meta-data repository to realize the warehousing process. For that, these tools are weak in meta-data evolution and reusability.

QELT takes advantage of its ability to achieve the automatic generation of data transformation from mapping guideline (mapping meta-data). It enables the generation of a set of procedures to be stored in the DBMS and to be called to refresh DW data. For that, it provides extraction from multiple sources by using SQL queries on the data sources. It uses meta-data to optimise the flow of these data. It reduces the update of the transformations process when meta-data rules evolve by an automatic generation of SQL transformation procedures. And finally, it offers the possibility to load data in the target DW database.

4.2 QELT Architecture:

QELT architecture (Fig.1) is very similar to the traditional warehousing tool. It integrates data from different and heterogeneous sources, applies transformation, and loads data in the DW database. Moreover, QELT is an active ETL; it interacts with meta-data to generate the transformations and to specify the loading with target schema. Hence, it optimises the flow of data and reduces the update of warehousing process by making automatic the creation of valid transformation. The order of the process is not conserved in QELT, for that it loads data inside DBMS in a temporary database, which has the same schema as the source data. After execution of the transformations procedures and creation of DW database, the temporary database will be deleted. The different components of QELT tool are described below:

4.2.1 The Meta-Data Components:

The Meta-data repository is the essential element in the system. It includes a set of information of which:

- The mapping meta-data (MM): this model is used to describe the mapping guideline and its mapping expression. By mapping expression, we mean the needed information to identify how a target field could be mapped from a set of source fields.
- The source model (SM) contains the model of the source data. It covers relational, object oriented and semi-structured data modelling [7], [6].
- The target model (TM) is similar to (SM); it describes the target data model. Moreover, it could cover the multidimensional model used by OLAP processing. [7], [6].

4.2.2 The Extraction Process

The role of this process consists in collecting needed data from different sources. These sources could be traditional databases, inner digital documents that are produced by applications in enterprise, legacy data or even web documents published by other partners. The meta-data repository supplies the extraction process with the needed parameters. If the data source is a traditional database, the extraction process will consist of queries stored in the meta-data repository. Other data sources need extra programs to extract data. Therefore, the main role of the extraction process consists of supplying the system of the needed data.

4.2.3 The Loading Process:

The second step consists of loading data to a temporary database. This database will be used later to execute transformation queries. The temporary database has the same structure of the source database. This implies the creation of a temporary database for each source. In the case of integration data flat files or XML documents, we create a temporary database having a simple structure or we use XML Schema description for XML documents.

The loader reads the description of the data sources (physical model) from the meta-data repository and creates the temporary database. The second goal of the loader consists on filling the temporary database with the data. In order to not fill the temporary database with unused data, which could have a dramatic influence on the DBMS, the extraction process should

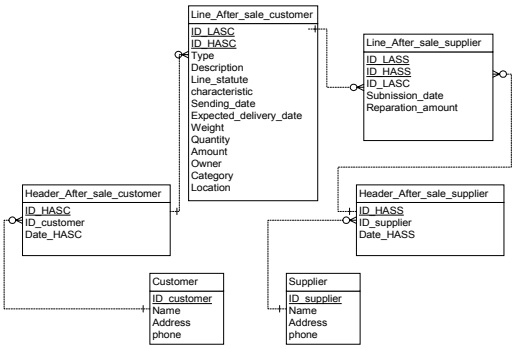


Fig 2. Target model for *After-sales service*

optimize the flow of data and not supply the loader with unneeded data.

4.2.4 The SQL Generator And Transformation Process:

The SQL Generator is a module, which can read useful parameters, rules, and mapping guideline from the meta-data repository to create an SQL transformation. Concretely, a query is performed on the mapping model (MM). The result file is used by the generator to produce the transformation queries. Then, a procedure is created in the DBMS containing these queries transformation. Indeed, the selection and filters get rid of superfluous data. During this process, all controls and check are applied to data. Thus, these queries are more than a cosmetic change for the data; it tackles the content, structure and valid values. We use SQL queries because they are easy to understand and are very efficient inside the DBMS. Existing tools should create a set of programs to achieve the transformation process; the optimization of these programs will not be easy. On the other hand, the SQL Generator could use the query optimization to get high performance.

The transformation process consists of the execution of these procedures including the full creation of the target database or the refresh of target data. The last statement of the transformation procedure includes the removal of the temporary database. We consider that the data transformation process is generated directly from meta-data. Thus, the designer will not be called to verify the consistency of the process, since this process uses mapping guidelines extracted from the meta-data repository. At the same time, the traditional request of meta-data is accessible for system's administrator. As we are using DBMS as an engine, we have tried to perform this functionality by generating the SQL procedure from meta-data.

5. CASE STUDY[1]:

In order to verify the effectiveness and efficiency of QELT, we held a case study to test samples with our tool. The case study applied QELT to data coming from a legacy source that was not designed to be SQL compatible since it includes VSAM files. This study focuses on the particular and difficult case of data coming from a legacy system.

[1] This case study has been held within TESSI Informatique, it included the development of a prototype of QELT.

The aim of this case study is to apply QELT in order to generate a DW and to use this warehouse to generate statistics and reports. Actually, realisation of each statistic costs approximately 5 days of development; the use of QELT reduces this time to be 2 days (including the analyse step). Without QELT complete *COBOL* programs under the mainframe computer have to be developed. The continuous demands of different types of statistics (the client's needs are never the same) pushes us to create the DW database in a different server by using QELT tool.

The data describes A*fter-Sales Service* for a group of department stores. The target model is defined in Fig.2. We have used *MS-SQL Server* as target the DBMS. Meta-data is created on the same server and it is used to help the warehousing process. It contains the source data description such as files structures, records, and field's descriptions. Meta-data also describes the mapping guideline between the source and the target. The used meta-data model is defined in Fig.3.
A prototype of QELT has been developed . This prototype uses the implementation of the model of the mapping guideline and the mapping expression (MM) to create automatically the needed transformation.

5.1 Realisation:

We proceed with three steps to realise our case study. These steps describe the ETL (Extraction, Transformation, and Load) process used to create data warehouses, but with a different order.

Step1: The Extraction Process

Since our data source is not SQL compatible, we used existing *Cobol* extraction programs that generate ASCII files from VSAM mainframe files. These programs are generic for many applications. This means that the extraction process uses a limited set of parameters received from meta-data. The resulting ASCII

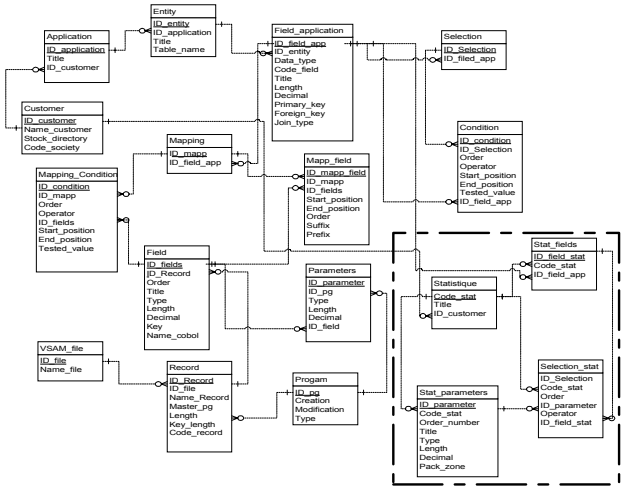


Fig 3. Case study meta-data model including mapping meta-data

files are transmitted by ftp to the DBMS server. We proceeded to limit the extracted files size to be around 100MB (effectively 108MB including 273,000 rows). This decision aims at not charging the DBMS server with temporary files all at once.

Step 2: The Loading process:

In order to load data to be accessible by the DBMS, we created a temporary database with a simple representation (a simple table). The structure of this table is identified by the meta-data (SM). A simple *Create Table* query was generated to achieve this process. After, we filled this database with the data existing in the ASCII flat files. We used *DTS(Data Transformation Server)* a module of *MS-SQL Server* to load the data. *DTS* enables the integration of data coming from sources such as flat files inside the DBMS.

Step3: The transformation process:

By using the QELT interfaces, we requested the creation of transformations. This request has been translated to a creation of a SQL Script within the DBMS including SQL statements. These statements perform data transformation and the creation of a new database (target DW). These statements have been generated from the representation of the mapping expression described in the mapping meta-data (mapping guideline). Moreover, these statements use a set of proper functions for the *SQL Server* such as the *CONVERT* function used to convert characters to integer, float, or date, *DATETIME* function, and *SUBSTRING* function, etc. The last statement of the procedure consists of removing *(DROP)* the temporary table and database.

Step4: The restitution process:

We used *Business Object*[2] (BO) to restitute data outside the target DW. This tool enabled us to generate easily statistics and needed reports.

5.2 Results:

The case study has presented the following results:

- At the extraction level: the *COBOL* extraction programs gave three files. The first and the second contain successively the customers and the suppliers. The third file contains all the rows needed to fill other tables. The case study covered 6% of the client's existing data, including 1 month of activities. The total size of the files was (108 MB). The storage of these files was temporary because it was going to be deleted after loading the temporary database.
- At the loading level: this process executed the creation of a temporary database query. We used *DTS* to load the data from files into the database. This process was very fast and took 25 seconds[3] for all the files.
- The generation of SQL transformation from meta-data was not a complicated process. These transformations have been stored as procedure in the DBMS. The execution of these transformations took 2 minutes, 15 seconds[3], including the creation of the target DW, execution of the mapping expression between source and target attributes, and the drop of the temporary database. The performance of QELT is good even without optimisation of the queries. As a matter of fact, to prove this functionality we tried to apply the same transformation with existing tools (*Data Junction* and *Sagent*), and the results were not at all better than those of QETL.

We summarize here, the trumps of QELT:

- *Efficiency*: QELT has proved good efficiency for the transformation process. Indeed, these transformations are executed by the DBMS having a good performance to execute SQL queries. Furthermore, the used procedure was not optimised with any query optimisation technique. We believe that using an optimisation technique for these queries will make QETL overtake other programs based existing tools.
- *Maintenance:* The active aspect of QELT architecture supports maintenance. Indeed, the system is able to generate automatically new transformations; no extra update will be needed to enable evolution. Thus, the administrator will not be called to create manually the transformation according to the mapping guideline. Moreover, the administrator will not be called to verify the consistency of the transformations and the target DW.
- *Improving Data Transformation:* QELT meta-data describes the mapping between source and target data. During the warehousing process, these mapping specifications are expressed by the mapping expression, which will be transformed to SQL statements. This ensures a full interaction between meta-data and transformation process.

6. CONCLUSION AND FUTURE WORK:

Enterprise data warehouse systems will evolve into federated structures. The next generation of data warehouses should be able to handle changing business requirements on a close to real-time basis, integrating data from web sources, etc.

Meta-data plays an essential role to create and maintain data warehouses, and it is affected by evolution of management rules. It is essential to develop tools and utilities with skills to hand the evolution directly to the warehousing process. The idea of this paper consists of defining a model for the mapping guideline and the use of this model to create an ETL tool. We investigated the importance of the mapping guideline for several applications. We illustrated the mapping expression with some known examples. We specified a model to represent mapping guideline and mapping expression.

Therefore, we tried to prove this model by the creation of a warehousing tool. This tool (QELT) is original in its capability to read the mapping guideline defined in the meta-data repository to create the transformation process. QELT is a query based ETL; it uses SQL queries to achieve the transformation between the source and the target. Thus, we use the DBMS with SQL queries as a transformation engine. We presented a case study using this tool to proof its effectiveness. Additionally, we used a meta-data repository containing the mapping guideline to generate automatically the transformation by using the SQL Generator.

Future work will be held on the meta-data level to create a more suitable and efficient model for describing meta-data of the data warehouse. Such issues were briefly studied in [3] and [4]. Extending these notions, investigating usability, and defining an ontology-based model will be the core of our future study.

7. ACKNOWLEDGMENTS:

We would like to thank the Research and Development team at TESSI-INFORMATIQUE that helps us to achieve this study including Andre Bonnevialle, Cedric Barlet, Christophe Roger.

[2] http://www.businessobjects.com

[3] Under Pentium III Processor 667 MHz and 128 MB RAM with Windows NT Server 4.0

8. REFERENCES:

[1] C.Quix: "Repository Support for Data Warehouse Evolution", proceedings of the International Workshop on Design and Management of Data Warehouses, DMDW'99, Heidelberg Germany 1999.

[2] E.Alsène, "The Computer Integration of the Enterprise", IEEE Data Engineering Management, Vol.46, pp.26-35 February 1999.

[3] J. Van Zyl, D.Corbett, Millist W. Vincent: "An Ontology of Metadata for a Data warehouse Represented in Description Logics", CODAS'99, Wollongong, Australia, March 27-28, 1999. Springer, Singapore, ISBN 9814021644, pp. 27-38.

[4] R.Rifaieh, N.A.Benharkat: "A Translation Procedure to Clarify the Relationship between Ontologies and XML Schema", 2nd Internet Computing Conference IC2001, Las Vegas, USA June 2001.

[5] S.Chaudhuri, U.Dayal, "An Overview of Data Warehousing and OLAP Technlology", ACM SIGMOD Record 26(1), March 1997.

[6] T.Stöhr, R.Müller, E.Rahm: "An integrative and Uniform Model for Metadata Management in Data Warehousing Environments", in proceeding of the International Workshop on Design and Management of Data Warehouse DMDW'99, Germany June 1999.

[7] Do, H.H; Rahm, E.: "On Metadata Interoperability in Data Warehouses". Technical Report 01-2000. Dept. of Computer Science, Univ. of Leipzig, March 2000.

[8] R.J.Miller, L.M.Haas, M.A.Hernandez: "Schema Mapping as Query Discovery". Proc. Of the 26th VLDB Conference, Cairo, Egypt, 2000.

[9] J.Madhavan, P.A.Bernstein, and E.Rahm: "*Generic schema matching with cupid*", In Proceedings of the 27th International Conferences on Very Large Databases, pages 49--58, 2001.

[10] M.C.Sampaio, E.M.F.Jorge, C.S. Baptista: "Metadata for an Extensible Data Warehouse Server". In proceedings of Workshop on Information Integration on the Web WIIW'2001, Rio de Janeiro, Brazil, pp.133-140.

[11] R.J. Miller, M.A. Hernandez, L.M. Haas, L.-L. Yan, C.T.H. Ho, R. Fagin, and L. Popa:" *The Clio Project: Managing Heterogeneity"*. SIGMOD Record, 30(1):78--83, 2001.

[12] M. Stonebraker and J. Hellerstein: "*Content integration for e-business".* In Proc ACM SIGMOD/PODS 2001 Santa Barbara, California, May 21-24, 2001.

[13] J.C.Grundy, W.B.Mugridge, J.G. Hosking, and P.Kendal: "Generating EDI Message Translations from Visual Specifications", In Proceedings of the 16th International Conference on Automated Software Engineering, San Diego, 26-29 Nov 2001, IEEE CS Press, pp. 35-42.

[14] M.Staudt, A.Vaduva, T.Vetterli: "Metadata Management and Data Warehousing", Technical Report May 99, The Department of Information Technology (IFI) at the University of Zurich.

[15] J.Berlin, A.Motro: "Database Schema Matching Using Machine Learning with Feature Selection". CAiSE 2002 Toronto, Canada, pp.452-466.