Towards a Matrix Based Approach for Analyzing the Impact of Change on ETL Processes

Ahmed Kabiri, Faouzia Wadjinny and Dalila Chiadmi

Computer Science Department, Mohammadia School of Engineers (EMI), Mohammed Vth University-Agdal
BP. 765 AV. Ibn Sina Agdal, Rabat, Morocco

Abstract

Extraction, Transformation and Loading (ETL) processes aim to extract data from data sources to targets, via a set of transformations. In many situations, an ETL process can be subject to changes for several reasons. For instance, data sources changes, new requirements and bug fixing. When changes happen, analyzing the impact of change is mandatory to avoid errors and mitigate the risk of breaking existent treatments. Several solutions have been proposed for this issue. We propose a new approach, based on matrices, for analyzing the impact of change on ETL processes. The goal of this paper is to present the theoretical fundament of our approach. We model ETL parts as matrices then we propose an algorithm which detects the affected parts in an ETL process, given an attribute deletion event. Compared with the existent solutions, our approach has the advantage of easiness.

Keywords: *Data warehouse, ETL processes, Conceptual modeling of ETL processes, Impact of Change***.**

1. Introduction

Extraction, Transformation and Loading (ETL) processes are responsible for the operations taking place in the back stage of data warehouse architecture. In a high level description of an ETL process, first, the data are extracted from data sources (databases tables, flat files, ERP, internet, etc.). Then, the extracted data are propagated to a special-purpose area of the warehouse, called the Data Staging Area (DSA), where their transformation, homogenization, and cleansing take place. Finally, the data are loaded to the central data warehouse (DW) and all its counterparts (e.g., data marts and views) [6].

It is widely recognized that building ETL processes is expensive in terms of time and money. It consumes up to 70% of resources [3], [5], [4], [2]. A set of studies have shown this fact [2]. At the building phase, the ETL designer defines how to map sources attributes to the targets ones. Depending on the quality of data sources, he has to specify how to clean and how to conform data. For example, he states how to remove duplicates and how to standardize attributes values. To assist the designer in his task, the research community has suggested several proposals for modeling of ETL processes [7], [8], [9], [10], [11], [12], [13], [14], [15], [16].

After the building phase, arrives the maintaining phase. In order to reply to new requirements, initiated by an involvement in the business or just to develop a bug fix reported by stakeholders, change cannot be avoided. Generally, change is neglected although it is a fundamental aspect of information systems and database [17]. Often, the focus is on building and running systems. Less attention is paid to the way of making easy the management of change in systems. Consequently, the time and the cost of maintaining ETL processes, already very expensive, are increasing more and more. Research community catches this need and supplies, in response, some solutions as we will see in related works section.

In this paper, we propose a new approach for analyzing the impact of change on ETL processes. Our goal is to simplify the maintaining phase. In our approach, we define a new formalism, based on matrices, to represent the ETL process parts. By applying matrices multiplication operations, we derive a m atrix, called K matrix, which overview the ETL process. We propose an algorithm, using the K matrix, to detect the affected parts in an ETL process given an attribute deletion event. This event can occur either in sources, in targets or inside the ETL process.

The remaining of this paper is organized as follows. Section 2 presents the change issue in ETL processes. Section 3 is dedicated to our approach. Section 4 presents related works. We conclude and present our future works in section 5.
In the rest of the paper, we don't make difference between ETL process, ETL job and scenario.

2. Change Issue in ETL

An ETL process integrates heterogeneous sources to a data warehouse. It can also be used as a flow router, where it supplies applications with data often as flat files. Therefore, it can be seen as a bridge between sources and

targets enriched with a set of transformations. Based on that, entities that may cause changes in ETL processes are sources, targets and transformations. Below, we recall and analyze some fundamental properties and aspects of these entities before tackling changes in ETL processes.

1) Sources. They encompass all types of data sources. The two famous types are databases and flat files. Recently, by the expansion of the internet, XML sources become spreading. More details and explanation about the characteristics of sources are available in [3]. In an ETL context, and regardless of their nature, sources are:

- Autonomous or semi autonomous. They are under the control of ope rational departments or they may be external to the entity exploiting them. Consequently, any change in sources is not systematically reported to ETL teams.
- Diversified. In the same process, an ETL can include various types of sources.
- Unstable. Their contents are constantly changing. In addition, the data structures or, broadly speaking, metadata may evolve.

2) Targets. Conversely to sources, they are controllable. The famous types are data warehouses (as databases) and flat files (in mediation context). But theoretically, it can be any kind of storing. The two last properties above (about sources) are also valid for targets.

3) Transformations. Transformation is a broad term, meaning all the data processing operations performed from sources to targets (mapping attributes and data flow). Such transformations include join, filter, sort, etc. operations. This is the critical phase in the whole process since it carries out the logic of business process instanced as business rules. Besides, from the first treatment to the last one, the structure of processed data is changing. The workflow schema is modified step by step, either by adding or removing attributes. Therefore, at this level, the most important and sensible parts are *business rules (*BR) and *flow structure*.

The three entities mentioned above may be the place of changes in ETL processes. It is interesting to note that, while changes in targets and transformations are scheduled, changes in sources are not under control. Therefore, it is difficult to have a 100% vision of when and where a change event can take place because of sources autonomy. But, does any change in the three entities above is meaningful to ETL? The answer is not obvious. It depends on several factors, which we synthesize below.

1. **The nature of change** (delete, add or modify).
2. **The location of change** (sources, targets or transformations). Adding a field in the target has a direct impact on the ETL process, since it should feed such information. While a new field in sources does not necessarily impact ETL processes.
3. **The implementation performed by the developer**. The manner of writing requests to extract data from sources influences the process stability. For example, selecting data via *select * from Source* is different from typing *select* A_1, A_2....A_n *from Source*. The first option is sensible to change, while the second one is immune to adding fields.
4. **The position of change inside data**. Adding a field at first rank differs from adding it at the end. Suppose we use a flat file, which has the structure $(A_1; A_2; \ldots ; A_n)$, as a source. Let's suppose that the attribute A_1 is involved in a business rule *BR* and the attribute A_n is free (not used). Let A_0 be a new attribute in the concerned source. Extracting data from $f1(A_0;A_1,A_2;....A_n)$ will lead to errors and may cause ETL crash. Indeed, during *BR* evaluation, ETL will consider A_0 as A_1. However, dealing with $f2(A_1;A_2,.....;A_n;A_0)$, leads to consider A_0 as a subpart of A_n.

The combination of the four factors cited above, gives an idea about the complexity of managing ETL changes. A trivial solution is to perform ad-hoc changes in ETL processes. But, this solution is detrimental to the success of the ETL project [3]. Therefore, we need to develop an approach which basic mission is to supply answer to the question: ***what is the impact of a given change on ETL process?*** An example of this will be ***what part of the ETL process is affected if we delete an attribute*** [2]. To answer this question, ***we need to know which attributes/tables are involved in the population of a certain attribute*** [2].
In the next section, we present our approach to meet this need.

3. Our Approach for Handling Changes in ETL Processes

In this section, we expose our approach, based on matrices, for handling changes and evolutions in ETL processes. To illustrate our proposal, our discussion we will be based on a running example presented in section 3.1. Then we expose the fundaments of our approach. Particularly, we show how to model ETL parts as matrices in section 3.2 while we build the K matrix in section 3.3. During this section, we consider sources and targets as flat files for concision and simplicity reasons. In section 3.4, we visit

databases resources which are more complex and more popular. Namely, we expand our proposal when sources and targets are tables in relational databases. Finally, in section 3.5, we suggest an algorithm for analyzing the impact of deleting an attribute on ETL process according to our approach.

3.1 A Running Example

Let's consider an ETL process called *job1*; reproduced in figure 1. *Job1*aims to populate a target O from a source I, having the following structures respectively: O(*CustKey; Name; ZipCode; City; Country; Address*) and I(*CustId; Fname; Lname; Town; Region*). Our scenario involves 3 transformations depicted in figure1: (1) flow standardization (*Format* box), then (2) flow validation (*Validate* box) and finally (3) flow conforming (*Conform* box). At the end, data is loaded to the target O.

Fig. 1 Design of job1 feeding target O from source I.

Figure 2 describes how the component *Format* handles transformations. It underlines how input fields are linked to output fields during *Format* step. The same mechanism (input fields attached to output fields via mapping rules or business rules) is reused in other transformation steps. However, extraction and loading steps are quite different.

	List of Input fields	Transformations	List of Output fields
I	CustId	I.CustId	CustKey
	Fname	concat(I.Fname, I.Lname)	Name
	Lname		
	Town	I.Town	City
		set to "Country_Name"	Country
	Region		

Fig.2. Mapping rules associated with the component *Format*

3.2 Matrix-Based Modeling of ETL Parts

In this section, we introduce a matrix-based representation of ETL parts. In our approach, all ETL parts are represented by matrices. For instance, figure 2, which depicts the behavior of *Format* component by mapping input attributes to output ones, can be represented by a matrix that we denote *F* (cf. figure 3). The rows and columns of *F* are the input attributes and the output attributes of *Format component* respectively. Consequently, the size of *F* is *card (inputs fields) ×card (output fields)*, where *card* refers to cardinality function.

The elements of *F* belong to {0, 1}. More precisely,

$$F[i,j] = \begin{cases} 1 \; ; & \text{if the output field ordered at column } j \\ & \text{involves the input field positioned at rank } i. \\ 0 \; ; & \text{otherwise.} \end{cases}$$

For example in figure 3, $F[1,1] = 1$ because the population of *CustKey* involves *CustId* (cf. figure 2). Since other output fields are not concerned, the first row of *F* (vector associated to *CustId*) is [1 0 0 0].

Similarly to matrix *F*, we build *V* and *C* the matrices associated to *Validate* and *Conform* components respectively. The figure 3 denotes them. In next section, we show how to combine them.

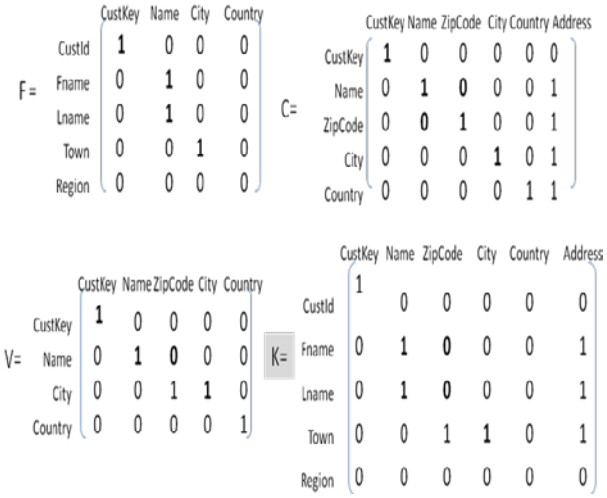

Fig.3. Related matrices to job1 components and it K matrix.

3.3 The K Matrix

In previous section we have shown how to build matrices *F; V; C.* Let *I* and *O* the representative matrices of the source and the target components involved in *job1* (cf. figure 1). According to this figure, the output fields of component I are the input of *Format* component. Stated in terms of matrices, the columns of *I* are the rows of *F*. Similarly the flow is propagated from a component to its successor. Thus the matrices product *I×F×V×C×O is*

valid and calculable[1]. We note the resulting matrix K as depicted in Eq. (1).

$$K = I \times F \times V \times C \times O \quad (1)$$

For *job1*, we have calculated the K matrix expressed by Eq. (1) when sources and targets are flat files (the matrices *I* and *O* are equals to the identity matrix). The final result is reproduced in figure 3.

As one can see, the rows of *K* are the attributes of the source I, while the columns of *K* are the attributes of the target O. Therefore, the ETL job presented in figure 1, which bridges the dataset I and the destination O, is equivalent to the K matrix.

Interpretation of the K Matrix: The *K* matrix summarizes the relationship between sources and targets at the field level. It supplies useful information.

Vertically, for a given output field *OFD*, it s hows the active input fields, which are involved in the population of *OFD*. For example, *ZipCode* vector shows that the input field *Town* is the only active attribute regarding *ZipCode* field. The *K* matrix shows also the autonomous output fields, which are independent of any input fields. They are derived during the definition of ETL process. According to matrix formalism, the elements of its associate vector are equals to zero. The field *Country* is an example of autonomous attributes, that are consequently, protected from sources changes.

Horizontally, vectors show the "customers" of each input field. They are attributes having the entries of the vector equal to 1. For example, the row of the attribute *Town* shows that it is used to populate three output attributes: *ZipCode, City and A ddress*. At the other extreme, rows having all entries equal to zero, underline inert input fields. They are unused anywhere in ETL process. Thus, any change in such fields is indifferent to ETL.

3.4 Extension to Databases

In previous section, we have considered the source and the target as flat files. In this section, we consider them as database tables because databases resources are more complex and involve special features. In addition, both of them are famous type of storing data.

[1] In matrix formalism, two matrices *A* and *B* can be multiplied ($A \times B$) only if the number of columns of *A* is equal to the number of rows of *B*.

So stated differently, we aim to build *K* matrix when sources and targets are database tables. To meet this need, we keep the graph of figure 1 and we extract data by the following request *REQ*.

REQ: Select CustId, Fname, Lname, Town from ***TableS*** *where* A_1*=param1 and* A_2*> param2 order by CustId, Fname, Lname.*

Param1 as *param2* are parameters of the ETL job. A_1 and A_2 belong to *TableS* although they are not propagated inside ETL job. They serve only to filter rows selected by *REQ*.

Now, let's construct *I* the representative matrix of the source component I (cf. figure 1) handling the above query *REQ*.

The rows of *I* are all attributes of *TableS,* while **the columns of *I* are** the selected attributes of *REQ* that are the input fields of the next component (component *Format* according to figure1). **The entries of *I*** (cf. figure 4) are defined as in previous section (whether the input field contributes to calculate the output attribute or not, the element takes value one or zero).

$$I = \begin{array}{c} \\ CustId \\ Fname \\ Lname \\ Town \\ A1 \\ A2 \end{array} \begin{array}{c} \begin{array}{cccc} CustId & Fname & Lname & Town \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad I_2 = \begin{array}{c} \\ CustId \\ Fname \\ Lname \\ Town \\ WHERE \\ ORDER\text{-}BY \\ Param1 \\ Param2 \end{array} \begin{array}{c} \begin{array}{cccc} CustId & Fname & Lname & Town \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$I_1 = \begin{array}{c} \\ CustId \\ Fname \\ Lname \\ Town \\ A1 \\ A2 \end{array} \begin{array}{c} \begin{array}{cccccccc} CustId & Fname & Lname & Town & WHERE & ORDER\text{-}BY & Param1 & Param2 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Fig. 4 *I* Matrix and its expanded matrices related to relational source I

Finally, the arrangement of the matrix *I* is depicted in figure 4. But, such modeling of *I* is restrictive. It does not allow control over all features of *REQ* other than select part. To overcome this situation, *we have to handle other parts by expanding I to* I_1 *and* I_2, following the relation Eq. (2).

$$I = I_1 \times I_2 \quad (2)$$

According to this relation, the rows of I_1 are those of *I* while the columns of I_2 are those of *I*. Obviously, the rows

of I_2 are the columns of I_1. As one can see in figure 4, the columns of I_1 catch all clauses of *REQ*. The elements of I_1 are defined similarly. Let's note that I can be seen as a sub-matrix of I_1. Specifically, I and I_1 are identical when *REQ* contains only select clause. In fact, we don't need to expand I in such situation. I_2 is constructed like identity matrix. Therefore, we can show that the elements of $I_1 \times I_2$ belong to {0, 1}.

In other side, the loading step is performed by an insert query which can be modeled as we did for *REQ*. Finally the K matrix of the ETL job depicted in figure 1, when sources and targets are database tables, is calculable following the formula Eq. (3). O_1 and O_2 are the expanded matrices of the target O:

$$K = I_1 \times I_2 \times F \times V \times C \times O_1 \times O_2 \quad (3)$$

3.5 Analyzing Impact of Change on ETL processes

The K matrix defined in last sections supplies useful information. In particular, it solves the first issue regarding "which attributes are involved in the population of a certain attribute". In this section we deal with the second issue regarding "What part of the ETL process is affected if we delete an attribute".

Before focusing on the impact of attribute deletion, let's note that this event can take place at 3 levels (source, target or in the intermediate step). It is clear that if we delete any D_j (output attribute), there is no need to keep all treatments related to D_j: The performance of the ETL job is increased by eliminating both, inert fields and archaic transformations. However, deleting an input attribute (A_j or B_j) is questionable. Indeed, when this event occurs, a decision should be made (by the administrator or the designer) whether:

a) To delete all parts associated to A_j; or
b) To substitute A_j by a default value; or
c) To extract equivalent information somewhere.

The option (a) is not practicable, while option (b) makes output data constants (regarding substituted fields). Therefore, the option (c) is more suitable, except that it will lead to deep rework. Absorbing changes is another issue that we omit in current paper. However, in all cases the detection of affected parts is mandatory.

In our approach, the underlying of the impacted parts of an ETL job, given an attribute deletion event, is supplied by the following algorithm.

1. **Inputs**: *Field-name, ETL-Job, Type*.
2. **Outputs**: List of Affected parts.
3. If *Type* is a source
4. **Begin**
5. {**S**} = get-concerned-sources (*Field-Name*)
6. For each ***s*** in {**S**}
7. Input = *Field-Name*
8. $\prod_{i=1}^{i=m} Mi$ = get-k-matrices-developed-form-for(***s***)
9. For (j=1; j < m+1; j++){
10. Output = get-linked-columns-to(Input, Mj)
11. List = List + add(Output, Mj)
12. Input= Output }
13. Clean-and-Format-then-Return (List)
14. **End**
15. Else, If *Type* is an intermediate
16. **Begin**
17. {**E**} = get-concerned-branches (*Field-Name*)
18. For each e in {**E**}
19. index = get-index-of-first-instance-as-out-Field (*Field-Name*)
20. Input = *Field-Name*
21. $\prod_{i=index}^{i=m} Mi$ = get-k-matrices-developed-form-for(e)
22.
23. For (j=index; j < m+1; j++){
24. Output = get-linked-columns-to(Input, Mj)
25. List = List + add(Output, Mj)
26. Input= Output }
27. Clean-and-Format-then-Return (List)
28. **End**
29. Else If *Type* is a target
30. **Begin**
31. {**T** } = get-concerned-targets (*Field-Name*)
32. For each ***t*** in {**T**}
33. target-field = *Field-Name*
34. $\prod_{i=1}^{i=m} Mi$ = get-k-matrices-in developed-form-for (t)
35. For (j=m; j > 0; j--){
36. If *Field-Name* is in rows of Mj Then {
37. Outputs= get-linked-columns- to (target-field,Mj)
38. List = List + add(Outputs, Mj) }
39. Else {
40. List = List + add(target-field, Mj)
41. Inputs= get-linked-rows-to (target-field,Mj)
42. Inputs = keep-inactive-fields-in(Inputs, target-field])
43. if Inputs is Null then stop
44. else target-field = Inputs
45. } }
46. Clean-and-Format-then-Return (List)
47. **End**

Algorithm Explanation:

1) Line 1 describes **the inputs of the algorithm**. *ETL-Job* specifies the given scenario concerned by the evolution. *Type* indicates the lactation where event takes place. It takes 3 possible values {source, target, intermediate}. *Field-name* indicates the attribute name subject of change.
2) Line 2 describes the **output of the algorithm**. It returns the list of affected parts of an ETL processes given a deletion event.
3) Line 3 to line 14 treats the case of **attribute deletion at source level**:

a) At line 5, for a given input field *Field-name*, we identify a set of sources ***{S}*** that contains *Field-name.* At line 7, the variable *Input* handles the subject of change. Obviously, at the beginning *Input* is *Field-name*. Then for each *s* in *{S}* we get a branch of K matrix in relation with *s* represented in developed form ($\prod_{i=1}^{i=m} Mi$).(line 8)

b) From line 9 to line 12, we parse each matrix belonging to the product $\prod_{i=1}^{i=m} Mi$ one by one in order to collect affected parts.

c) At line 10, *get-linked-columns-to(Input, M_j)* function parses the vector of the field *Input* in matrix M_j. It gets columns having elements = 1. The result is stored in the variable *Output.*

d) At line 11, we update the variable *List* by adding the variable *Output* collected previousely and M_j which sympbolzes the compmoent where the change has taken place.

e) Too at line 12, we update variable *Input*. The input of M_{j+1} is affected fields in M_j.

f) Finally, at line 13, the function *Clean-and-Format-then-Return(List)* works on *List* result by eliminating redundant information. Then it formats the result *List* in order to display a final list of affected parts by component.

4) Line 15 to line 28 treats the case of **attribute deletion at intermediate level**. This bloc is similar to those of source bloc except that the starting point is not the source but somewhere inside of ETL scenario. Exactly at the first definition of involved attribute. This task is performed by the function cited at line 19.

5) Line 29 to line 47 treats the case of **attribute deletion at target level.** In that case, we need to go back until the first derivation of the attribute which has been deleted:

a) At line 31, for a given input field *Field-name*, we identify the set of targets ***{T}*** that contains *Field-name.* Then for each *t* in *{T}* we get it K matrices represented in developed form ($\prod_{i=1}^{i=m} Mi$) and in relation with *t*.

b) At line 35 we parse each matrix belonging to the product $\prod_{i=1}^{i=m} Mi$ one by one in order to collect affected parts.

c) At line 36, for a certain matrix M_j we checks if *Field-name* is an input field in M_j.

d) If this condition is true (line 37 – line 38), we update variables and go to next matrix. It is identical to line 11, 12 (cf. 3.c and 3.d).

e) Else (*Field-name* is not among input attributes of M_j) this is the first derivation of *Field-name* during ETL process. Since we delete *Field-name*, line 40 to line 44 aims to see, in ulterior treatment, if involved attributes in the population of *Field-name* during M_j step ar e useful or not. If used elsewhere we keep them, otherwise we eliminate associated treatment.

f) Finally, at line 46, the function *Clean-and-Format-then-Return(List)* works on *List* result by eliminating redundant information. Then it formats the result *List* in order to display a final list of affected parts by component.

Illustration: The algorithm above deals with "What part of the ETL process is affected if we delete an attribute". To make it clear, we consider *job2*, an ETL process depicted in figure 5. In *job2* each component is labeled ***X.Y***, where ***X*** indicates its representative matrix and *Y* is a free text for describing the meaning of the component.

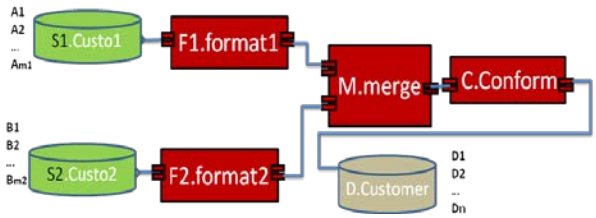

Fig. 5 Design of *job2* feeding Customer Dimension.

By applying the same procedure presented in section 3.3 and section 3.2, we get the K matrix of *job2* and the matrix of each component. Explicitly, we can show that:

$K_1 = S1 \times F1 \times M \times C \times D$ for the branch of the source S_1.
$K_2 = S2 \times F2 \times M \times C \times D$ for the branch of the source S_2.
With slight abuse of notation, we note[1]: $K = K_1 + K_2$.
Or in detailed form:

$$K = K_1 [A_1 ... A_{m1}; D_1 ... D_n] + K_2 [B_1 ... B_{m2}; D_1 ... D_n].$$

Below, example1 illustrates attribute deletion at source level A_j from S1. While example2 illustrates attribute deletion at target level D_j from the destination customer.

Example 1: In reference to figure 5, let's delete A_j from S1.

Thus, $K_1 = S1 \times F1 \times M \times C \times D = \prod_{i=1}^{i=m} Mi$ (m = 5) and {**S**} = {S1.custo}.
To simplify we consider $S1 = D = I$ (identity matrix) and $(K_1, F1, M, C) = (K, F, V, C)$ represented in figure 3. Let's

[1] We use the sum operator for composition. When $m_1 = m_2$, the sum of K_1 and K_2 is possible in matrices theory, but even, it is insignificant since the input attributes of K_1 and K_2 are not identical

suppose that A_j is *Town* field. Therefore, *Clean-and-Format-then-return* function in the above algorithm, will display the affected BR regrouped by component. The message seems like "Deleting attribute *Town* impacts:

- BR of *City* attribute at *F1-foramt1* component.
- BRs of *City* and *ZipCode* attributes at *M.merge* component.
- BRs of *City*, *ZipCode and Address* attributes at *C.conform-and-check* component".

Example 2: Now, suppose we delete the output field *Address*. We can see in matrix *C* (cf. figure3) that the attribute *Address* does not appear in its rows while the participant's fields (*Name, ZipCode, City,* and *Country)* to populate the target field (*Address*) are used elsewhere. Therefore, the attribute *Address* is initiated at *C* matrix. Consequently, the impact is only at "BR of the attribute *Address* at *C.conform-and-check* component".

4. Related Works

Design and Modeling: Research community enriches the field of conceptual modeling of ETL processes with several approaches [8],[9],[10],[11],[12],[13], [14], [15], [16],[7].These proposals differ on formalism and technology used. *But they have the same drawback: no support and functionalities to deal with changes in ETL processes*.

Evolution: A report on evolution and change in data management is available in [17]. The authors present a summary of issues in relation with the topic. Furthermore, a categorization of issues is given. Zooming on the area of data warehouse, change can be either in scheme or in the data saved from the first population of the DW. Managing data evolution, contrarily to schema evolution, over time is a basic mission of DW. Consequently, research efforts in DW evolution are oriented to schema versioning. Thus, in [18], the authors present an approach to schema versioning in DWs. Based on graph formalism; they represent schemata parts as a graph and define algebra to derive new schemas of DW given a change event. The formulation of queries invoking multiple schema versions is sketched. Same authors rework their proposal in [19] by investigating more data migration. X-Time [20] is a prototype of these efforts. *However, let note that these proposals deal with evolution only in DW. Stated in ETL words, they focus on targets side. Conversely, the proposal [21] deals with changes in DW sources*. In this work, the authors abstract all parts of DW, particularly ETL activities, as a sequence of enriched queries modeled by graphs. The graphs are annotated (by the designer) with actions to perform in response to change event. An algorithm to readapt the graph, given an evolution event in sources, is supplied. *However, this approach is difficult to implement, because of enormous amount of additional information required in nontrivial cases* [1]. Thus, the authors extend their work in [22] proposal, especially deep explanations of the above algorithm were given and the prototype architecture is presented too. This effort has been implemented via HECATAEUS prototype [22], [23]. This tool aims mainly to enable what-if analysis and regulation of relational database schema evolution.

Our work offers broad scope of managing changes in ETL setting. Indeed, it is not restrictive to evolutions in one side or any type. Our approach takes into account, in one view, changes that occur either in sources or targets or inside the ETL process. At high level, it detects affected components or affected steps and then it underlines mapping rules affected. Besides, matrices conversely to graph, offer an easy look over ETL.

5. Conclusion and Future Works

ETL processes are famous with two tags: complexity and cost. Also, evolution in such environment cannot be escaped. Consequently, it is mandatory to have a helpful tool and an effective approach for simplifying the maintenance task of these processes.

In this paper, we have presented our approach for handling impact of change analysis in ETL processes. Our approach is based on matrices. Especially, we have represented ETL parts as matrices and we have shown how to derive a new matrix called K matrix, by applying multiplication operations. We have exposed too, how K matrix summarizes the relationship between the input fields and the output fields and how it synthesizes the attributes dependency. Particularly, the K matrix tells us "which attributes are involved in the population of a certain attribute" and which attributes are the "customers" of a given one. Finally, we have proposed an algorithm to detect affected part of ETL job when a change deletion event occurs, either in sources or targets or inside ETL.

In future works, we plan to extend the scope of events managed by taking into account *Add* and *Modify* events at the attribute level. Another way to advance this work is to investigate on building, automatically, matrices related to an ETL process and to look into how to absorb change events. Still, exciting challenge will be the enhancement of our model. Thus; it is more interesting to catch further details, when defining association between attributes, instead of restricting to binary {0, 1} relationship. Given

that, ETL optimization according to our approach will be attracting.

References

[1] A. Dolnik, "ETL evolution from data sources to data warehouse using mediator data storage". In MANAGING EVOLUTION OF DATA WAREHOUSES MEDWa Workshop, 2009.

[2] A. Simitisis, P. Vassiliadis, S. Skiadopoulos and T. Sellis, "Data warehouse Refreshment", Data Warehouses and OLAP: Concepts, Architectures and Solutions, IRM Press, 2007.

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

[4] W. Inmon, D. Strauss and G. Neushloss, DW 2.0 The Architecture for the next generation of data warehousing, Morgan Kaufman, 2007.

[5] M. Golfarelli, "Data Warehouse Life-Cycle and Design". In Collection of Encyclopedia of Database Systems, 2009, pp. 658-664.

[6] P. Vassiliadis and A.Simitsis, "EXTRACTION, TRANSFORMATION, AND LOADING", http://www.cs.uoi.gr/~pvassil/publications/2009_DB_encyclopedia/Extract-Transform-Load.pdf

[7] M. Bouzeghoub, F. Fabret and M. Matulovic-Broqué, "Modeling Data Warehouse Refreshment Process as a Workflow Application", in Design and Management of Datawarehouse (DMDW) Workshop, 1999.

[8] P. Vassiliadis, A. Simitsis and S. Skiadopoulos, "Conceptual modeling for ETL processes". In 5th ACM Int Workshop on Data Warehousing and OLAP, pp. 14-21, 2002.

[9] A. Simitsis and P. Vassiliadis, "Methodology for the conceptual modeling of ETL processes". In CAISE Workshops, 2003.

[10] A. Simitsis, "Mapping conceptual to logical models for ETL processes". In the 8th ACM Int.Workshop on Data Warehousing and OLAP, 2005, pp.67-76.

[11] D. Skoutas and A. Simitsis, "Designing ETL processes using semantic web technologies", in the 9th ACM Int.Workshop on Data Warehousing and OLAP, 2006, 67-74.

[12] D. Skoutas and A. Simitsis, "Ontology-based conceptual design of ETL processes for both structured and semi-structured data", International Journal. on Semantic Web and Information Systems, Vol. 3, No. 4, 2007, pp. 1-24.

[13] Z. ElAkkaoui and E. Zimanyi, "Defining ETL Workflows using BPMN and BPEL". in the 12th ACM Int. Workshop on Data Warehousing and OLAP, 2009.

[14] L. Muñoz, J. Mazón and J. Trujillo, "Automatic Generation of ETL processes from Conceptual Models". in the 12 th ACM Int. Workshop on Data Warehousing and OLAP, 2009.

[15] J. Trujillo and S. Lujan-Mora, "A UML Based Approach for Modeling ETL Processes in Data Warehouses", in Proceedings of ER, 2003, 307-320.

[16] M. Golfarelli, "New Trends in Business Intelligence", in proceeding of 1st International Symposium on Business Intelligent Systems (BIS'05), 2005, pp. 15-26.

[17] J.F. Roddick et al, "Evolution and Change in Data Management - Issues and Directions", SIGMOD Record 29, 2000, Vol. 29, pp. 21-25.

[18] M. Golfarelli, J. Lechtenborger, S. Rizzi and G. Vossen, "Schema Versioning in Data Warehouses". in ER Workshops 2004, LNCS 3289, pp. 415–428.

[19] M. Golfarelli, J. Lechtenborger, S. Rizzi and G. Vossen, "Schema versioning in data warehouses: Enabling crossversion querying via schema augmentation", Data and Knowledge Engineering, 2006, pp.435–459.

[20] S. Rizzi and M Golfarelli, "X-time: Schema versioning and cross-version querying in data warehouses", in International Conference on Data Engineering (ICDE), 2007, pp.1471–1472.

[21] G. Papastefanatos, P. Vassiliadis, A. Simitsis and Y. Vassiliou, "What-If Analysis for Data Warehouse Evolution", in DaWaK conference, 2007, LNCS 4654, pp. 23–33.

[22] G. Papastefanatos, P. Vassiliadis, A. Simitsis and Y. Vassiliou, "Policy-Regulated Management of ETL Evolution", Journal on Data Semantics XIII, LNCS 5530, 2009, pp. 146–176.

[23] G. Papastefanatos, P. Vassiliadis, A. Simitsis and Y. Vassiliou, " HECATAEUS: Regulating Schema Evolution. Data Engineering", in International Conference on Data Engineering (ICDE), 2010, pp. 1181-1184.

A. KABIRI Engineer degree in Computer Science in 2003. PhD student in Computer Science. Main research interests: Data integration, Data warehouses, ETL, MDA. Practitioner in Business Intelligence field.

F. WADJINNY Engineer degree in industrial computing in 1992; DESA degree in computing Networks and Multimedia in 2002; PhD in Computer Science in 2010 with distinction; Professor of computer science at ITSMAERB, Rabat/ Morocco; 12 recent publications papers (between 2008 and 2011); Ongoing research interests: Mediation systems, P2P data integration, Data warehousing, Information systems security.

D. CHIADMI is a Professor of Computer Science at Mohamadia engineering School, Mohammed V Agdal University, Morocco. She received her Diploma (M.Sc.) and her Doctorate (Ph.D.) in Computer Science. She is Director of SIR Laboratory (Information System and Distribution Laboratory), Mohammed V Agdal University since 2005 and Head of the Research team Intégrale (INTEGRAtion de ressources d'information sur Le wEb). She is Editor in chief of eTI journal (www.revue-eti.net), a digital open access journal on IT, and she served on various program committees of conferences and journals in computer sciences. Her research interests include semantic data and services integration, services composition, and BI. Book, book chapter and about 20 recent papers published.