USERS EXPECTATIONS FEEDBACK CONSISTENCY AS A FIRST STEP FOR A BETTER DATA QUALITY

[1]L. BRADJI , [2]M. BOUFAIDA
LIRE Laboratory, Mentouri University of Constantine, Algeria
E-mail: [1]bradjilouardi@yahoo.fr, [2]mboufaida@umc.edu.dz

ABSTRACT

User feedback is a new direction employed to assist users in building data integration systems and to improve their quality. We claim that the first step to the efficient data quality improvement in a data integration system is the consistency of user's expectations feedback because user(s) can have multiple expectations of the quality of the value and the content of the query results at the same time. Therefore, in this paper we propose an approach that allows one to detect user's expectations quality problems, especially inconsistency, and so their improvement before their use for determining the data quality issues at the data sources in the second step. This approach takes a set of queries from the log file and the user's expectations relative to the results of each query to identify and prioritize the inconsistencies of user's expectations values. Thus, to accomplish this, three algorithms are proposed. The first algorithm manages the user's expectations inconsistency related to the same query. The second algorithm divides queries into groups of similar attributes by using the clustering technique of data mining. As the rerun of the queries is time consuming, the approach takes samples of data to determine the queries results interactions. The quality of these samples is determined in advance in order to ensuring the correctness of queries results interactions. The third algorithm takes both the queries results interactions and the user's expectations to address the inconsistency between the user's expectations of queries results interactions. To detect the user's expectations inconsistencies, we have established a set of user's expectations axioms and inference rules, which are used by the algorithm to deduce an automatic expectation value. This algorithm introduces a solution, which is inspired from the axioms systems of Logic, to solve the inconsistency.

Keywords: *Data Integration Systems, Data Quality, Consistency, User Feedback, User Expectation, Query Results, Clustering, Quality Of Value, Queries Interactions.*

1. INTRODUCTION

Data integration is a critical and fundamental element in a variety of technologies, including data warehouses, business intelligence (BI) applications, service-oriented architectures (SOA), master data management (MDM) applications, and data-centric architectures [1].

Data integration manages the process of combining data residing in different autonomous sources for users to submit queries and providing unified queries results (views) [2, 3]. It has become the center of broad theoretical work, and numerous open problems remain unsolved. One of the most pertinent problems that users are facing is the quality of data. Much works has been done on query processing and choosing plans under cost criteria. However, not so much is known about incorporating data quality management into data integration systems [2].

Data quality management is a crucial part of any data integration process. It may be considered the first step to the integration process, as quality data is the key to achieving profitable insights. The data integration analysis will not be successful until good data quality processes are in place.

To facilitate data integration, some databases researchers have explored the use of feedback solicited from users [4]. User feedback is a growing theme in data integration systems as allows the use of the user's expectations to guide the building of data integration systems and/or to improve the quality of the services they provide [5]. Therefore it is quite logical that the user feedback should be taken into account to evaluate and improve the quality of data integration systems. This happens for three main reasons. First, data quality systems that consider all possible data quality problems during the query processing are difficult to write. Second, a fully automated solution that meets the quality requirement is not always attainable [6, 7].

Third, users can be the genuine collaborators, rather than merely sources of data [8]. Consequently the quality of data offered (query results) in the integration systems may or may not meet the user's expectations. These reasons imply that, in general, a portion of data quality problems has to be detected and/or (possible) corrected after the query processing by the users. Hence user's expectations should be involved in the data quality improvement strategies.

One effective mechanism to take users expectations feedback is to manage the consistency of the user's expectations values relative to queries results in a comprehensive and continuous data quality improvement program for the data integration systems in order to be exploited to improve the quality of data integration systems.

In this context, we suggest an approach for the detection, analysis and improvement of user's expectations feedback quality problems as a first step towards improving the quality of data integration system from the user's perspective. Let us notice that this work deals with the expectation relative to the quality of value and content of the queries results in order to enhance the quality of data sources. The main idea behind this proposal is that the data quality can't be assessed independently from the users who use data [9].

This paper is structures as follows. In section 2 we give a state of art regarding the data integration systems and data quality. Section 3 discusses some works related to the user feedback mechanism. In Section 4, we firstly motivate our approach and we also give its objectives, follows by presenting the most proposed concepts and definitions, and its general principle. Section 5 describes in detail the main algorithms of our approach. Finally, Section 6 shows our conclusions and draws some perspectives.

2. BACKGROUND

In this section we review the concepts that are used throughout this paper. We firstly discuss some existing approaches for data integration systems. We also recall the data quality principles.

2.1. Data Integration Systems

Integration can be enabled via a flexible query answering system that accesses multiple sources on-demand or via a data warehouse that pre-assembles data for known or anticipated uses [10].

Two main approaches to data integration can be identified: Virtual data integration and Materialized data integration. Virtual data integration is a method for centralizing data without physically consolidating it first (data reside only at sources). Materialized data integration is a method where the unified view of data is materialized, for instance, in a data warehouse [11, 12]. The proposed work can be applied in both approaches.

The basic components of a data integration system are wrappers and mediators. A wrapper is a software module that accesses a data source, extracts the relevant data, and presents such data in a specified format, typically as a set of relational tables. A mediator collects, cleans, and combines data produced by wrappers and/or other mediators, according to a specific information need of the integration system. The specification and the realization of mediators is the core problem in the design of an integration system [3, 13, 14].

2.2. Data Quality

The data quality is often defined as "fitness for use", i.e. the ability of integrated data to met user expectations [9, 15]. This implies that quality of data is seen as relative and subjective. Data considered appropriate for one use may not possess sufficient quality for another use [16],

Data quality is multi-dimensional concept and in the data quality literature several frameworks providing categories and dimensions as way of facing data quality problems [2, 15, 22].

Data quality problems may be divided into two main categories: problems regarding data coming from one source and problems regarding data coming from multiples sources. Both main categories may be further divided into two subcategories: data quality issues on instance and data quality issues on schema level [17, 22].

This work focuses particularly on the instance-level where data quality issues become very significant and then can strongly affect query processing in data integration systems [11].

3. SOME RELATED WORKS

Early, the most data quality researches works focused mainly on developing techniques for querying multiple data sources and building large data warehouses [10, 22, 23].

A substantial number of research works and projects have been conducted on the topic of feedback of user's expectations. However, there is still a lack of specific proposals for the data quality in data integration systems that consider the users expectations and tools that put these proposals into practice. The most dimensions of the expectations perspective identified in the literature are: Privacy, Content, Quality of Values, Presentation, Improvement and Commitment [9].

Although the user feedback based method is very costly with respect to the time required [18], its incorporation in several automatic tasks has shown to be useful and powerful. In [5], the authors propose a solution to incorporate the end-user feedback into information extraction and integration programs. The authors of [18] present a system for implicit user feedback using rough set theory to improve the quality of metasearching. In [19], the authors explored the use of feedback supplied by end users for annotating, selecting and refining schema mappings in the context of dataspaces.

Existing research works have shown that taking into account user feedback as a first class citizen presents several advantages for the construction and improvement of data integration systems. However, in practice, user feedback has be only seen as annotations that a user provides to comment on artifacts of a data integration system, be they matches, mappings, integration schema, queries or queries results [4, 5, 18, 19]. Therefore, theses works have study the user feedback conflicts at the semantic level. They have identified different kinds of feedback and proposed set of terms used for annotating objects on which feedback is given. For the annotation of values of the content and quality of value perspectives of the query results, they have proposed three terms: true positives (when the result meets the user’s expectations), false positives (when the result do not meets the user’s expectations) and false negatives (when the user’s expectations are not returned) [4, 5].

In the context of data cleaning, some works have integrated the user feedback in automatic data cleaning process in order to involve a user at any intermediate result produced by data transformation to improve manually the quality of poor data [6, 7]. The problem is that, when using data cleaning tools, intermediate results obtained during the cleaning process are not expected or evaluated by the users which can generally only expect and/or evaluate the final result. Besides, these works do not allow the management of user’s expectations feedback and their quality.

The consistency and validity of user’s feedback have been studied in [4] where only the inconsistency of user’s expectations feedback of the same query is considered. This work is based on the principle that the inconsistencies in feedback emerge from changes in user’s expectations. Our approach considers also the changes at the data sources and takes into account the queries results interactions to deal with inconsistencies of multiple queries.

4. AN OVERVIEW OF PROPOSED APPROACH

This section contains material and concepts that form the basis for our approach and also describes briefly its fundamentals tasks.

4.1. Motivations and Objectives

As the data must meet the user requirements, the user’s expectations feedback should be considered during the build and improvement data integration systems. The first step in each data integration programs quality is the management of user’s expectations feedback consistency relative to the queries results in order to detect the data quality problems.

Currently most user’s expectations feedback works are dedicated to the study of consistency of user’s expectations relative to the same query with the consideration of the user is the origin of the expectations changes. Therefore, the keys motivations for our proposition are:

(1) Theoretical foundations for consistency. As the consistency is powerful part of the logic, there is a need of theoretical foundations based on logic for the user’s expectations feedback consistency. In this work, we have deal with this limitation by establishing an axioms system based in the use of a set of user’s expectation rules inference and axioms to infer automatically expectation value from existing user’s expectations. The axiom system in logic and the set theory in mathematics are the basis of our theoretical foundations for user’s expectations consistency. For more details of these both theories, see [21, 23].

(2) Data sources changes. When the data sources are updated, consequently the user’s expectations can be changed. Then it is necessary to take into account the time of modification of data sources in the validation of user’s expectations. For example two user’s expectations of the same user for the same query results with data sources updates can be different. In this case there are not inconsistencies between these expectations.

(3) Query results interactions. There are, generally, interactions between query results. However, none of the proposed works for a user feedback consistency takes into account these interactions. Therefore this work differentiates between user’s

expectations consistency of the same query (mono query) and expectations consistency between queries (multiple queries). Let us give an example motivating this reason: if the result of a query A is a subset of the result of a query B, then if B meets user's requirements then B also.

This work considers a relational context; specifically, it deals with the user queries. It addresses the problem of the quality of user's expectations by exploiting the axiom systems, set theory and clustering techniques of data mining. It concerns the quality of values and content of queries results.

4.2. General principle of our approach

The principle of our approach is to capture the different user's expectations via a user interface and to apply clustering techniques to both queries and user's expectations that are representative of data integration system usage in order to deduce the inconsistencies of the user's expectations instances. As shown in figure 1, this proposal uses a list of queries which are extracted from the log file, and a set of the user's expectations to identify inconsistencies and correct them if it is possible. The proposed approach has three mains steps for the management of consistency: Mono query consistency, results queries interactions clustering and multiples queries consistency. The sampling, queries interactions and query processing steps are used to help the multiple queries results interactions process and therefore, in this paper, we present them as part of the multiple queries consistency step.

The mono query consistency step uses an algorithm called Mono Query results Consistency (MoQC) algorithm to chek the inconsistency of user's expectations of the same query results for each query and then selects an optimal solution to detect and eliminate inconsistency if it exists. The multiple queries consistency step uses an algorithm called Multiple Queries results Consistency (MuQC) algorithm to verify the consistency between queries results and correct them if it is necessary. The basis of this algorithm is a set of user's expectations axioms and rule inference build by applying the principles of axiom systems and set theory to the queries results interactions. The clustering of queries is realized by an algorithm called Queries Interactions Clustering (QIC). The QIC computes the queries similarity which is measured by the similarity of attributes between queries. The clustering method proposed in this approach is inspired from data mining-based warehouse performance optimization approach presented in [20]. As the queries interactions don't imply the queries results (views) interactions, the QIC algorithm takes samples of high quality from each of the base tables of data sources to determine the interactions that may exist between queries results.

Let us notice that our approach can process in batch mode and/or in online mode.

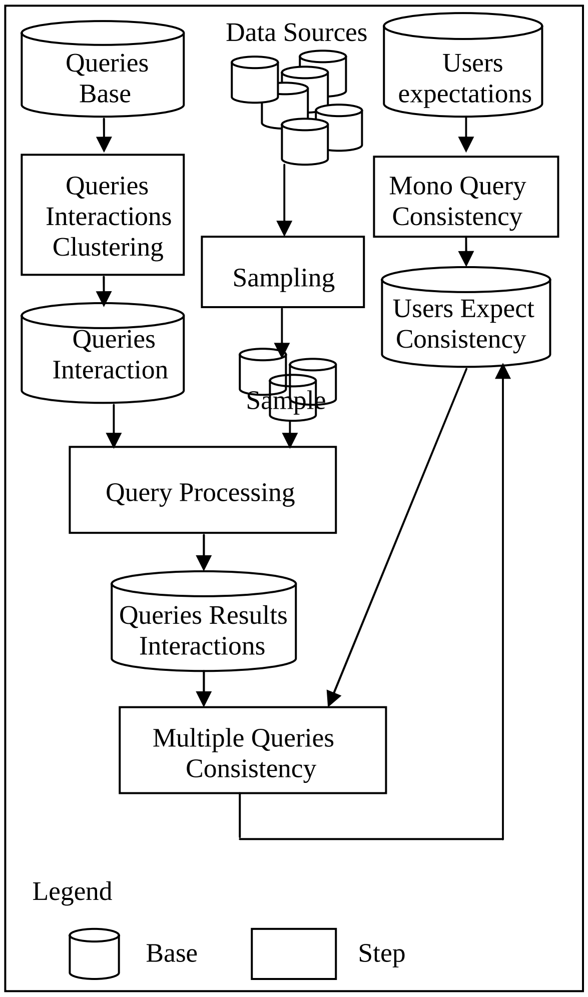

FIGURE 1: USERS EXPECTATIONS CONSISTENCY APPROACH

4.3. Basic Concepts of user expectation feedback

In this subsection, we define the concepts used in this paper according to our approach.

User Expectation

The user expectation can be defined as a user evaluation of the quality of value and content of the query result. We define three values for the user expectation: High (H), Poor (P) and Unknown (X). When the query results meet the user expectations, then the user expectation value is high. If the query results don't meet the user expectation, then the expectation value is poor. Contrary, the expectation value is unknown if the user can't or don't evaluate the query result (it considers as missing or incomplete value).

Users Expectations Matrix

In order to manage the feedback of user's expectations, we reference the instances of these last in a matrix whose the rows are vectors (noted UE, for User Expectation) that characterize the feedback of the user expectations for given query Qi. Each vector is a quintuplet defined as follows:

$$UE=(Q_i, User(j), T_{exp}, T_{update}, UE(k,j))$$

The User(j) design the user which evaluate the quality of query result of Q_i. The T_{exp} is the time of evaluation of the quality of the result of Qi by user(j). The time T_{update} is a set of couple (A_n, T_n) where A_n is attribute of Qi and T_n denote the time of last modification of A_n. This T_{update} which is not considered by currently works is useful for the management of consistency of user's expectations because changes in data sources imply changes in user expectations. Let us notice that the term same query result signifies the same query with the same T_{update}. The UE(k,j) is the k^{th} user expectation value of a user j for the query results i.

Origins of user's expectations inconsistencies

As we have indicated above, the user's expectation changes are linked to three main factors: users, data sources updates and queries. For the user factor we distinguish two cases: multiple users have different expectations for the same query results at the same time and the same user have multiple expectations over time for the same query results without data sources updates. Similar to user factor, we distinguish two cases of query factor: same query or multiple queries. Therefore we have identified four situations where the inconsistencies of user's expectations instances can be occurred:

Situation 1: Same user and same query
Situation 2: Multiple users and same query
Situation 3: Same user and multiple queries
Situation 4: Multiple users and multiple queries.

As one user can give multiple expectations to the query results, then same or multiple users mean the same: multiple user's expectations feedback. For this reason, our approach deals with inconsistencies in two phases (algorithms).

Before explaining algorithms relative to each phase, we give firstly the concepts which we have introduced to identify the consistencies of users expectations feedback.

Concept 1: Users expectations consistency.

If there are different user's expectations feedback of a given query results with the update time of data sources (T_{update}) is the same, then the user's expectations feedback are consistent. Let us notice that a single value of user's expectations of each query results will be computed from the multiple user's expectations values of this query results in UE. This value is stored in a query results-attributes matrix.

Concept 2: Users expectations Interpretation.

When the user performs the query, the system creates automatically instances in the user expectation matrix. We define this user expectation as an interpretation (i.e. the expectation value is unknown).

Concept 3: Users expectations model.

When the user put the value of the user expectation of the query results then we define it as model (i.e. the value is high or poor)

Concept 4: Positive consistency.

User's expectation feedback of query results Q is positive consistent (noted posit_consist(Q) or h+) if each value of the user's expectations feedback of this query results in UE is high. Therefore the posit_consist(Q) value is high.

Concept 4: Positive inconsistency.

Users expectation feedback of query results Q is positive inconsistent (noted posit_inconsist(Q) or p+) if each value of the users expectations feedback of this query results in the UE is poor. Therefore the posit_inconsist(Q) value is poor.

Concept 4: Negative consistency.

Users expectation feedback of query results Q is negative consistent (noted negat_consist(Q) or h-) if the number of the users expectations feedback with the value is high (noted m) is higher than the number of the users expectations feedback with value is poor (noted n). Therefore the negat_consist(Q) value is high.

Concept 4: Negative inconsistency.

Users expectation feedback of query results Q is negative inconsistent (noted negat_inconsist(Q) or p-) if n is higher than m. Therefore the negat_inconsist(Q) value is poor.

Journal of Theoretical and Applied Information Technology
15th November 2011. Vol. 33 No.1
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The basic idea behind the categorization of consistency is that the positive consistence and positive inconsistence will be used to proof the negative consistency (resp. inconsistence) of queries results. The user expectation model concept is crucial also because only evaluated query results (high or poor) are considered for the consistency management.

Let us notice that in our approach, the inconsistency exists if and only if there are two expectations such that one is high and the other is poor for the same query at the same T_{update}. If one of the expectations is unknown, the system states only the value of the other expectation valid. This is available for each situation.

5. User's expectations feedback Consistency management

Although the user's expectations changes are linked to three mains factors: users, data sources updates and queries. The proposal approach is based in the query result factor that gathers the three above factors. Because the query result confirms that the query is performed by a given user in multiple data sources. Therefore the approach deal with inconsistency in two cases: mono-query results and multiple-queries results.

Note that the Queries Base (noted QB) is extracted from the log file which listing every request made to the data sources. The system constructs the List Queries matrix (noted LQ) from the UE.

5.1. Mono Query results Consistency

In this step, the system takes the UE and produces an Users Expectation Consistency matrix (noted UEC) in which each query results have only a single user expectation value (positive consistence (h+), positive inconsistence (p+), negative consistence (h-) or negative inconsistence (p-)). This task is realized by the MoQC algorithm. Let us notice that the MoQC takes into account only the users expectations models. As shown in figure 2, the MoQC deals with the two cases of user factor: same user and multiple users.

Same user

In order to avoid the inconsistency in advance, the user can insert, modify and delete expectations via a user interface. But if MoQC identifies multiple user expectations at the same T_{update}, then the last user expectation only will be state valid. But when the user expectations have different T_{update}, the system validate true the both user expectations because the inconsistency as we have indicated above occurred when user's expectations concern the same query results at the same time of modification of data sources.

Multiple users

It is quite logical that multiple user's have different expectations (at the same T_{update}). But the question is: what a user expectation will be taken? In MoQC, we propose three solutions for this situation and a developer needs to choice the appropriate one and/or combine them (solution 3). These solutions are:

1. Acceptation of each user's expectations.
2. Ignoring user's expectations.
3. Intervention of an expert to choice the correct user expectation or that of the user more qualified.
4. Selection of optimal user expectation value. This solution which we give it top priority consists to calculate the users expectation of given query results Q (UE.value(Q)) from the users expectations models in UE.

```
m=0 and n=0
For each Qi ∈ LQ (list_queries) do
  Repeat
  If UE.value(Qi)=High then Qi.m=Qi.m+1
  Else Qi.n=Qi.n+1
  End if
  Until end_of_(LQ)
  If same-user(Qi) then Take last user expect(Qi) Else
    If m=0 then posit_inconsist(Qi)  Else
      If n=0 then posit_consist(Qi) Else r=m/n
       If r > h then negat_consist(Qi)  Else
         If r < p then negat_inconsist(Qi)
         Else (r < h and r > p) Select solutions (1, 2, 3)
         End if
       End if
    End if
  End if
End if
End for
```

FIGURE 2: USER EXPECTATION CONSISTENCY MONO-QUERY ALGORITHM

The algorithm start by computing the two values m and n from UE for given query results. Then the algorithm calculates the ratio (noted r) that measures the quality of the existing value user expectations to infer a single user expectation value. This ratio is defined as follows:

Journal of Theoretical and Applied Information Technology
15th November 2011. Vol. 33 No.1
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$r = \frac{m}{n}(h, p),$$
$$h \text{ et } p \text{ are two thresholds and } h \geq p$$

The thresholds h>=1 (for high) and p<=1 (for poor) are used to will be compared to the ratio r and consequently to determine the user expectation value.

As the third solution is complex to perform because it requires the rerun of the query processing which can be time consuming, we give priority to the solutions 1, 2 and 4. As shown in figure 2, the ratio r can be between h and p, and then the user during the configuration of the algorithm will indicate the solution to be taken in this case. The user expectation value produced by the MoQC is stored in the UEC.

5.2. Query results interactions detection

Before applying the MuQC algorithm, it is necessary to determine the queries results interactions which is hard and does not take into account by existing approaches. Our approach has established the QIC algorithm especially for the queries clustering. The QIC clusters the queries into groups in Queries Attributes Matrix and then performs the queries of each group on the sample of data in order to determine the queries results interactions.

Query results-attributes matrix

It is similar to the query-attribute matrix used for the materialization of views and selection of index in data warehouse (see [20]). Differently to this approach which selects only the attributes that may support materialized or indexes for each query, our approach selects only the attributes in query results and the attribute of user expectation value (noted Q-Attrib). The user expectations value is extracted from the UEC (noted UE.value). For example for Q2 in figure 3, Q-Attrib= {a1, a5, a6, UE}.

Let QR(Qi,Aj) be a Queries Results-Attributes(noted QR) matrix where Qi and Aj design respectively a given query and a given attribute of given table of a given data source.:

$$QR(Q_i, A_j) = \begin{cases} 1 & \text{if } Aj \text{ is attribute in Query result of } Qi \\ UE.value & \text{if the attribute is the UE of } Qi \\ 0 & \text{otherwise} \end{cases}$$

Queries interactions clustering

The principle of this part of the QIC algorithm is to apply a clustering technique on LQ and UEC matrixes to construct the QR matrix and then the groups of queries are building by the application of clustering methods. The clustering is based in that the queries of the same group are similar in some attributes. Figure 4 shows an example of QR matrix corresponding to the example of queries used in [20] for view and index selection in figure 3. The queries Q1, Q2 and Q3 are used to select data that fulfill specified criterions from tables F, D1, D2 and D3.

Query 1: Q1

SELECT F.a1, SUM(F.a2) FROM F, D1 WHERE F.a1 = D1.a3 AND D1.a4 < 2000 GROUP BY F.a1

Query 2: Q2

SELECT F.a1, F.a5, AVG(F.a6) FROM F, D1, D2 WHERE F.a1 = D1.a3 AND F.a5 = D2.a7 AND D2.a8 = 'ABC' GROUP BY F.a1, F.a5

Query 3: Q3

SELECT F.a1, F.a9, SUM(F.a2) FROM F, D1, D3 WHERE F.a1 = D1.a3 AND F.a9 = D3.a10 GROUP BY F.a1, F.a9

FIGURE 3: EXAMPLE OF QUERIES [20]

	a1	a2	a3	a4	a5	a6	a7	a8	a9	UE
Q1	1	1	0	0	0	0	0	0	0	h+
Q2	1	0	0	0	1	1	0	0	0	p+
Q3	1	1	0	0	0	0	0	0	1	h-

FIGURE 4: SAMPLE QUERIES RESULTS ATTRIBUTES MATRIX

The main substeps of our queries interactions algorithms are:

Queries clustering. Figure 5 shows the part of the QIC algorithm that clusters queries stored in QR. After the construction of QR matrix, the algorithm affects queries to their groups in queries_clusters matrix. The clustering method is unsupervised because the number of groups is a priori unknown. Each group (line) is identified by an attributes-centric (noted Attrib-C) which is set of the most frequent attributes between the queries of the same group. The Attrib-C is calculated incrementally. When query is affected to group, then the Attrib-C is updated by the function Higher (see figure 5) which allows calculating the list of most frequent attributes in the group. To affect query to their groups (query can be affected to multiple groups), the algorithm measures query similarity (noted Q-Simil) between each Attrib-C group and the Q-Attrib of given query and then decides of its affectation or creation of new group, and so updates the Attrib-C,

As shown in figure 5, the function intersect is used to verify if Q-Attrib of given query has intersection with a given group. This function helps Higher function to find the more frequent attributes.

Query results interactions. Generally the query result is virtual and it is hard to determine the interaction between query results. In our study, we prefer sampling data because it is less expensive than reprocessing queries. It allows also using a data of high quality free from errors. Consequently, we ensure the quality of queries results interactions.

```
Function intersect (Q, G)
Let Q : Query and G : set of queries={Qj}
    Intersect = false
    J=1
    While not (intersect) do
      If Q-Attrib(Q) $\cap$ Q-Attrib(Qj)$\neq\emptyset$ then
          Intersect=true
      End if
      Next j
    End while
End function
For each Qi of queries attributes do
  Affectation=false
    For each group Gj do
      Q-Simil(Gj, Qi)=Attrib-C(Gj) $\cap$ Q-Attrib(Qi)
        If Q-Simil(Gj, Qi)$\neq\emptyset$ then
          Affect Qi to Gj
          Affectation=true
          Attrib-C (Gj) =Higher (Q-Attrib(Qi $\in$ Gj))
        End if
    End For
  If not (affectation) then
    Create new group G in queries cluster
    Attrib-C(G)=Q-Attrib(Qi)
    Affect Qi to G
  End if
```

FIGURE 5: CLUSTERING ALGORITHM

Once the queries results interactions are determined, the system performs the MuQC algorithm.

5.3. Multiple Queries results Consistency

After determination and validation of each value user expectation feedback of each query results by the MoQC algorithm, the system will be checked their consistencies by comparing them to the users expectations value inferred by using the queries results interaction and therefore doing the corrections if is necessary.

The existing approaches for user feedback consistency don't take into account the interactions that may exist between queries results which can supply inconsistency in user's expectations feedback.

In this perspective, this step deals with this problem by the MuQC algorithm which is based in the use of an axiom system proper to the user's expectations feedback domain.

Before the performing of the MuQC algorithm, the system will be firstly identified the queries results interactions by the QIC algorithm.

As each system is a set of axioms and inference rules, we have established the user's expectations rules inference by applying the inference rules of Modus ponens of Logic to the queries results interactions.

Firstly we remind the principle of the two rules:
Let p and q: two formulas:
Modus Ponens: if ($p\rightarrow q$) and p are true then q is true.
For more details of this rules, see [21].
In our case the value true (resp. poor) is equivalent to high (resp. false).

Queries results interactions.

In our work, the queries results interactions are concluded from the queries interactions matrix. We define these last as follows:

Let Resul(Q) design the query result of Q. then the interaction between two queries results of Qi and Qj (noted Q_interaction(Qi, Qj)) exist if

$$Resul(Q_i) \cap Resul(Q_j) \neq \emptyset$$

We distinguish three cases:

$$\mathrm{Resul}(Q_i) \cap \mathrm{Resul}(Q_j) = \begin{cases} \mathrm{Resul}(Q_i) \ \text{ i.e. } \mathrm{Resul}(Q_i) \subseteq \mathrm{Resul}(Q_j) & \textbf{(1)} \\ \mathrm{Resul}(Q_j) \ \text{ i.e. } \mathrm{Resul}(Q_j) \subseteq \mathrm{Resul}(Q_i) & \textbf{(2)} \\ \mathrm{Res} \neq (1) \text{and } (2) & \textbf{(3)} \\ \qquad \text{i.e. } \mathrm{Res} \subset \ \text{i.e } \mathrm{Resul}(Q_i) & \\ \qquad \text{and } \mathrm{Res} \subset \mathrm{Resul}(Q_j) & \end{cases}$$

This definition is important because the operator subset ($\subset$) in set theory is equivalent to the operator imply ($\rightarrow$) in Logic. Therefore by applying the rules inference to the queries results interactions, we verify and eliminate the user's expectations inconsistency.

Rules users' expectations inference

Before defining rules necessary to the MuQC algorithm, we will first indicate that rule infer only values of the user expectations of a given query results of Q (don't infer rule). Let A=Resul(Qi), B=Resul(Qj) and C=Res. Then the set of user's expectations axioms and rules (not all) established are:

Axiom1: If UE(Q) is High then UE(Q) is positive consistency (h+) xor negative consistency (h-).
Axiom2: If UE (Q) is Poor then UE(Q) is positive inconsistency (p+) xor negative inconsistency (p-).
Rule1: If A$\subseteq$ B is true and B is posit_consist then A is posit_consist.
This rule is logic because as A is subset of B and B is positive (all B is true) then A is high.
Rule2: If A$\subseteq$ B is true and B is posit_inconsist then A is posit_inconsist.
Rule3: If A$\subseteq$ B is true and A is poor then B is posit_inconsist xor negat_inconsist xor negat_consist.
Rule4: If A$\subseteq$ B is true and A is high then B is negat_inconsist xor negat_consist xor posit_consist.
Rule5: If A$\subseteq$ B is true and C is poor then B and A are posit_inconsist xor negat_inconsist xor negat_consist.
Rule6: If A$\subseteq$ B is true and C is high then B and A are posit_consist xor negat_inconsist xor negat_consist.

When applying Rules, if the user expectation value of A inferred is different of the user expectation value in UEC and QR, then the algorithm replaced it by the inferred value in the both matrixes UEC et QR. Let us notice that the principle of the MuQC algorithm is the use of positive consistent (resp. inconsistent) calculated and/or validated by MoQC algorithm (i.e. the negative consistent (resp. inconsistent) values can't used by above rules because they are fuzzy truth) to verify and correct the negative consistent (resp. inconsistent).

The problem of multiple rules can infer different users expectations for a given query results has been taken into account by the MuQC algorithm. The MuQC deals with this problem by applying the solution that we have given to it top priority in situation 4 (Multiple users and same query) but there is only one threshold (h=p). Therefore the inferred value is negative consistent or negative inconsistent.

Let us give an example:

Let: posit_consist(Qj) (1), negat_inconsist(Qi) (2) have calculated by the MoQC algorithm.
By applying the MuQC algorithm, we have concluded that: Resul(Qi)$\subseteq$ Resul(Qj) (3).
Then
By applying Rule1 to (1) and (2), we obtain:
posit_consist(Qi) (4).
Therefore (2) and (4) are contradictories.
According to MuQC, the user expectation value of Qi may be replaced by negat_consist(Qi).
Justification: During the application of the mono query algorithm we have choice the negat_consist(Qi) because the number of high is higher than number of poor which can be false reason. Consequently the verification of queries interaction consistency by the MuQC allows (only) the correction of errors doing during the MoQC algorithm for the queries that have the value of user expectation is negative consistency (resp. inconsistency).

6. CONCLUSIONS AND SOME PERSPECTIVES

Based on the idea that the use of domain knowledge is crucial to improve the quality of data, we claim that the user's expectations feedback in data integration system plays a central role in improving the quality of data integration system (data sources, processes and views). Currently several research works have addressed the problem of integrating the user feedback in data cleaning process. However they remain to a difficult problem, because there is no easy way for humans to provide feedback into such data quality program. Therefore the efficient method to the integration of user's expectations feedback in the data integration system is the implementation of system that allows the collection of users expectations and their management automatically in order to improve the quality of the data integration system. In this perspective, this work presents the first step for the end describes above which is the management of the inconsistency of the user's expectations feedback. Firstly, the system deals with the consistency of the users expectations of each query by using the algorithm of mono query results consistency. Secondly it computes the queries results interactions by applying the clustering methods and then it verifies the inconsistency between queries by using the multiple queries consistency algorithm. This algorithm is based on the use of user's expectations axioms and rules inference which is inspired from the axiom systems of logic.

As next step, we plan to apply the clustering method to the both queries results interactions and users expectations consistency matrixes to determine the data quality problems in the data integration system. We should classify the data quality problems according to the data integration proprieties with their priorities and then cluster them onto groups according to their problems classification. The implementation of a system that supports this approach is also ongoing work.

REFRENCES:

[1] R.Y. Wang, V.C. Storey and C.P. Firth, " A Framework for Analysis of Data Quality Research", *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, Vol. 7, N0. 4, AUGUST 1995 pp. 623-640.

[2] M.C.M. Batista, A.C. Salgado, "Information Quality Measurement in Data Integration Schemas", Proceedings of THE Fifth Workshop on Quality in DataBases, QDB, at the VLDB conference, September 23, 2007, pp. 61-72.

[3] V. Peralta, R. Ruggia, Z. Kedad and M. Bouzeghoub, " A Framework for Data Quality Evaluation in a Data Integration System",

[4] K. Belhajjame, N.W. Paton, A.A.A. Fernands, C. Hedeler and S.M. Embury, "User Feedback as a First Class Citizen in Information Integration Systems", Online Proceedings of Fifth Biennial Conference on Innovative Data Systems Research, Asilmor, CA, USA, January 9-12, 2011.

[5] X. Chai, B.Q. Vuong, A. Doan, and J. F. Naughton, "Efficiently incorporating user feedback into information extraction and integration programs", *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, Providence, Rhode Island, USA, June 29-July 2, 2009, pp. 87-100.

[6] H. Galharads, A. Lopes and E. Santos, "Explicitly Involving the User in a Data Cleaning Process", Technical report : DI-FCUL-TR-2010-03, Department of Informatics of the University of Lisbon, Faculty of Sciences.February 14, 2011.

[7] H. Galharads, A. Lopes and E. Santos, "Support for User Involvement in a Data Cleaning", Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'11), Toulouse, France, August 29-September 2, 2011.

[8] "Principles for Best Practice in Clinical Audit", National Institute for Clinical Excellence (NHS). Published by Radcliffe Medical Press Ltd, Abingdon, UK, 208 pp., 2002.

[9] A. Caro, C. Calero and M. Piattini, "A Portal Data Quality Model for Users and Developpers", Proceedings of the 12th International Conference on Information Quality (ICIQ), MIT, Cambridge, USA, Novembre 9-11, 2007, pp. 462-476.

[10] S.E. Madnick, R.Y. Wang, y.w. Lee and H. Zhu, "Overview and Framework for Data and Information Quality Research", ACMJournal of Data and Information Quality, Vol. 1, No. 1, June 2009.

[11] C. Batini, and M. Scannapieca, "Data Quality: concepts, methodologies and techniques", Springer-Verlag Berlin Heidelberg, 2010.

[12] J. Lie Yong Koh, "Correlation-Based Methods for Biological Data Cleaning", PHD thesis, National University of SINGAPORE, Malaysia, 2007.

[13] D. Calvanese, L. Dragone, D. Nardi, R. Rosati and S. M. Trisolini, "Entreprise modeling and Data Warehousing in TELECOM ITALIA", in *Journal of Information Systems*, Vol. 31, No. 1, 2006, pp. 1-32.

[14] V. Ferraggine, J.H. Doorn and L.C. Rivero, "Handbook of Research on Innovations in Database Technologies and Application: Current and Future Trends", IGI Global, Information Science Reference, 2009, pp. 1-1124.

[15] M. Angélica Caro, C. Calero, I. Cabellero and M. Piattini, "Data Quality in Web Applications: A state of the art", Proceedings of IADIS, 2005, pp. 364-368.

[16] D.D. Fehrenbacher, and M. Helfert, "An empirical research on the evaluation of data quality dimensions", Proceedings of the 13th International Conference on Information Quality, MIT, USA, Cambridge, November 14-16,2008, pp. 230-245.

[17] R.K. Kumar, and RM. Chadrasekaran, "Attribute correlation-Data cleaning using Association rule and clustering methods", *International Journal of Data Mining & Knowledge Management Process (IJDKP)* Vol.1, No.2, March 2011, pp. 22-32.

[18] R. Ali nd M.M.S. Beg, "User feedback based Metasearching using rough set theory", Proceedings of the 2008 International

Conference on Information & Knowledge Engineering, IKE 2008, July 14-17, 2008, Las Vegas, Nevada, USA. CSREA Press 2008, pp. 489-495.

[19] K. Belhajjame, N. W. Paton, S.M Embury, A.A.A. Fernandes and C. Hedeler, "Feedback-Based Annotation, Selection and Refinement of Schema Mappings for Dataspaces", Proceedings of 13th Conference on Extending DataBase Technology, EDBT, Mars 22-26,2010, Lausanne, Switzerland, pp. 573-584.

[20] K. Aouiche and J. Darmont, "Data Mining-based Materialized View and Index Selection in Data Warehouses", Journal of Intelligent Information Systems, Springer, VOL. 33? No 1, 2007, pp. 65-93.

[21] R.M. Smullyan, "First-order logic", Dover Publications, Inc., New York, 1995.

[22] L. Bradji and M. Boufaida, "knowledge Based Data Cleaning for Data Warehouse Quality", Proceedings of the International Conference, ICDIPC, July 7-9,2011, Ostrava, Czech Republic, CCIS, Vol 189, No. 2, pp. 373-384, Springer Verlag.

[23] L. Bradji and M. Boufaida,"Rules-based Data Warehouse Quality for Data Mining", Absract Proceedings of EURO'09, July 5-8, 2009, Bonn, Germany.