ICIT 2015 The 7th International Conference on Information Technology
doi:10.15849/icit.2015.0078 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

Quality Driven Approach for Data Integration Systems

Mohamed Samir Abdel-Moneim
College of Computing and Information Technology
Arab Academy for Science Technology & Maritime Transport
Cairo, Egypt
moh_samir_86@hotmail.com

Ali Hamed El-Bastawissy
Faculty of Computer Science
MSA University
Giza, Egypt
aelbastawissy@msa.eun.eg

Mohamed Hamed Kholief
College of Computing and Information Technology
Arab Academy for Science Technology & Maritime Transport
Alexandria, Egypt
kholief@aast.edu

***Abstract*—By data integration systems (DIS) we mean the systems in which query answers are instantaneously mapped from a set of available data sources. The query answers may be improved by detecting the quality of the data sources and map answers from the significant ones only. The quality measures of the data in the data sources may help in determining the significant data sources for a given query. In this paper, we suggest a method to calculate and store a set of quality measures on data sources. The quality measures are, then, interactively used in selecting the most significant candidates of data sources to answer user queries. User queries may include the user preferences of quality issues. Quality-based approach becomes increasingly important in case of big number of data sources or when the user requires data with specific quality preferences.**

Keywords*—*data integration; quality measures; data sources; query answers; user preferences

I. INTRODUCTION

Data Integration (DI) is the process of finding and retrieving data located at multiple locations, and allowing the user to view these data through a single unified view called global or mediated schema [1, 2]. Users use the global schema to pose their queries to a data integration system. The global schema provides a uniform access to data stored in heterogeneous and autonomous sources. The user no longer needs to consider which sources are relevant to their queries, how the data are structured at the sources, how to access the data sources, nor does he need to consider how to combine the results from different sources. Data integration system (DIS) queries the data sources according to the location of the required data:

- The required data may be found at a single source. In this case, the system has to query only that particular source.
- The required data may be found at many sources. In this case, the system may choose to query multiple sources or to query the best sources and combine the results.
- The required data may be scattered across many sources. In this case, the system must query different sources and combine the result from each source.

Different architectures for data integration systems have been proposed, but broadly speaking, most systems fall between warehousing and virtual integration [3]. In the data warehouse system, data from different homogeneous or heterogeneous sources are loaded into a physical database called warehouse through a process called extract, transform and load (ETL) so that queries over the data warehouse can be answered as shown in “Fig. 1”.

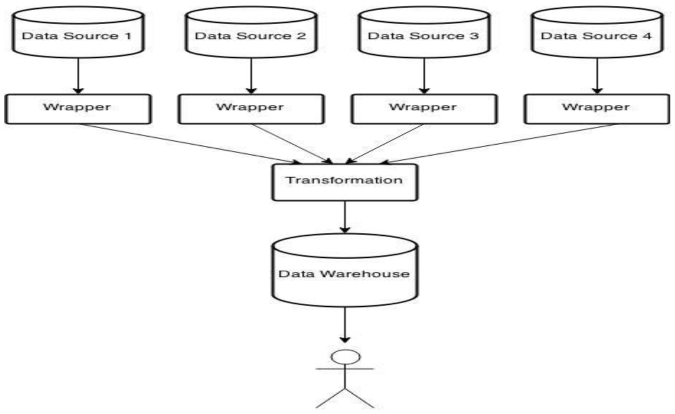


Fig. 1. ETL process

In virtual integration system, data remain in the sources and are accessed as needed at the query time. Data integration system (DIS) is often built as a mediator-wrapper architecture [4] as shown in "Fig. 2". Although the two approaches are different, many problems are shared across their architectures.

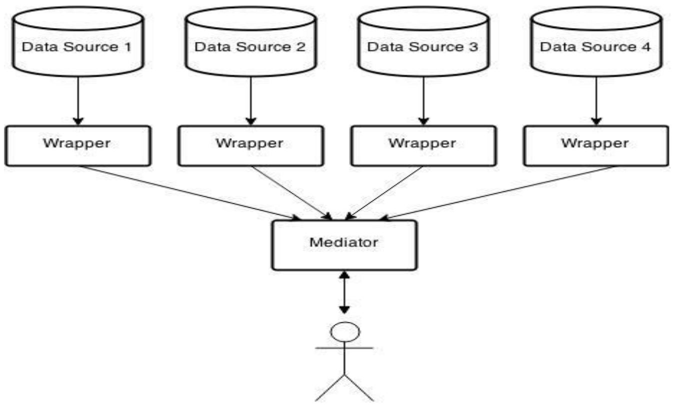


Fig. 2. An architecture of a data integration system

The quality of the data sources can dramatically change as data may be incomplete, inaccurate or out of date. In fact, the quality of the result depends mainly on two factors: the quality of the data at the data sources and the manipulation process that generates the resulting data from the data sources. Because of the high number and high diversity of participating data sources as well as their autonomy, it is important to store some quality-related measures to take it into consideration during query planning. Data Quality (DQ) has become very important in organizations and many application domains [5, 6]. DQ is based on a set of dimensions such as timeliness, completeness, and accuracy.

In this paper we present an approach that incorporates data quality into data integration systems in order to get satisfactory query plans. Our approach is based on adding quality system components such as data quality acquisition to be parts of any data integration system. We integrate attribute values from different data sources based on quality measures and user's preferences. We use quality measures to deliver query answers with satisfactory quality.

The rest of this paper is organized as follows. In Section II, we briefly discuss data quality dimensions for data Integration. Section III describes the work related to quality based data integration systems. We illustrate the architecture and functions of our data integration quality system components in section IV. Section V describes our quality driven query processing algorithm. We evaluate and validate our approach in section VI. The conclusion and future work are presented in Section VII.

II. DATA QUALITY CRITERIA FOR DATA INTEGRATION

Broadly defined, data quality means "fitness for use" [7, 8]. Therefore, the interpretation of the quality of data item depends on the user's needs. While some data quality may be appropriate for a given task or user, it may not be appropriate for another user or another task. Data quality dimensions depend on each other and only a suitable set of dimensions is appropriate for a given task. To decide which data quality dimensions to use, Wang and Strong [9] have empirically defined fifteen data quality dimensions considered by end users as the most significant. Wang and Strong classify these dimensions into contextual, intrinsic, representational and accessibility quality as shown in "Fig. 3".

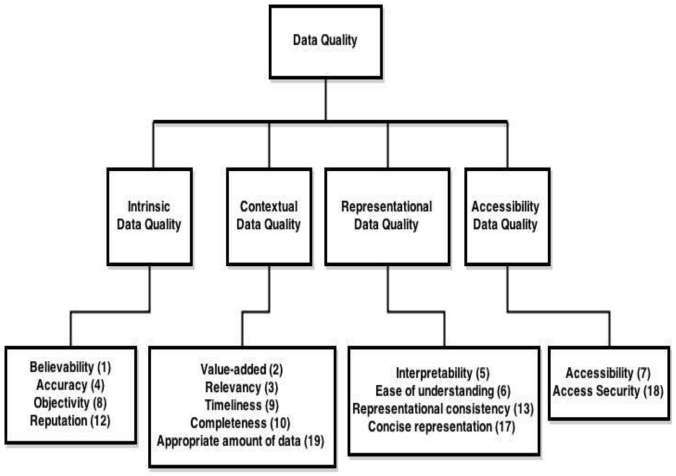


Fig. 3. A conceptual framework of data quality

The measurements of data quality dimensions can be done at different granularities:

1) Data source level: determine the quality for the whole source. Quality measures of this type remain unchanged as long as the source doesn't dramatically change.
2) Relation level: determine the quality of a relation in a data source.
3) Attribute level: determine the quality of an attribute in a relation.

In this we only focus on data quality dimensions that could affect the data integration process and could be considered important from user's prospective. We illustrate these dimensions as follows:

A. Accuracy

Several definitions have been defined for the term accuracy. Wang and Strong [9] defines accuracy as " The extent to which data are correct, reliable, and certified free of error". Redman [10] defines accuracy as "the degree of correctness of a given collection of data". In general, two types of accuracy are considered important by literature, syntactic and semantic [11]. Increasing accuracy of the query answer is important from user's prospective as data sources might contain incorrect or misspelling data.

B. Completeness

Wang and Strong define completeness as "the extent to which data are of sufficient breadth, depth, and scope for the task at hand" [9]. One of the main objectives of integration is to increase completeness. Completeness is one of the most important data quality dimensions in the integration of data sources. Querying one data source gives a set of results. Querying another data source gives another overlapping set. As the number of data sources queried increase, the result will be more complete.

C. Cost

Cost is the amount of money required for a query. Some data sources may charge users for accessing their data. Considering cost is important so that users can choose between free and commercial data sources.

D. Response Time

It is the amount of time when the mediator submit a query and receive the complete response from the data source. Users usually prefer data sources that have low response time. Response time is important in order to determine the time-outs and unavailability of data sources. Users waiting a long time for a response are more willing to terminate the query. Response time also could be one of the factors for source selection when a data integration system decides which data sources to query in order to answer a query.

E. Timeliness

Timeliness is how old the data are in a data source [12]. Timeliness in the context of data integration is the time between the last verification or update of the data and now. Timeliness is important as some data sources might be outdated and the user might be interested in getting up-to-date data.

III. QUALITY BASED DATA INTEGRATION SYSTEMS

In this section, we present an overview of research projects that have been proposed to perform query processing based on data quality. We focus on how they measure and store data quality, how they process queries and user interference option.

A. The DaQuinCIS architecture

The DaQuinCIS project [13] is designed to improve data quality in cooperative environments.

DaQuinCIS uses metadata to store the quality measures, the interpretation of the quality measures, and information related to the measurements.

DaQuinCIS follows global-as-view (GAV) approach for processing queries. DaQuinCIS decomposes queries submitted over a global schema to queries against local data sources. The query processing approach adopted by DaQuinCIS to find an answer to a query is structured as following:

1) A user posed a query Q on the global schema.
2) The query Q is then decomposed according to the schema mapping that maps each concept of the global schema in terms of the local sources. Therefore, the query Q is unfolded to Q1,…Qk queries to be sent over the local data sources.
3) Executing queries Q1,….Qk, returns results R1,….Rk. A record matching algorithm is used to find items common to both results.
4) The final result is built according to the following rules:
   a) If no quality constraint is specified, the result is generated by selecting the best quality values.
   b) Whether there are quality constraints, the result is generated by examining whether the constraints satisfy the whole result.

B. Data Integration Techniques based on Data Quality Aspects

Gertz and Schmitt [14] are used data quality to develop data integration techniques within an object oriented data model and used a metadata to store the information about data quality. Quality dimensions such as accuracy, completeness and timeliness are selected for the purpose of database integration. Gertz and Schmitt have also developed query language extensions to be used for specifying data quality goals for global queries and in data integration.

If objects are conflicted semantically, the object with the best data quality must be chosen. If conflicts exist between the integrated objects but they are different at their quality level, then these objects need to be grouped in order to rank the results.

Regarding user contributions in the integration process, the user has less flexibility in determining priorities of the quality dimensions. Because the data quality are offered as the most up to date or the most accurate and not offered in weights or percentages. Consequently, users will not be satisfied by combinations of quality priorities. One result might satisfy a user for a particular task, but of poor quality for other. Also, the user has no option if he wants to integrate more than data source to find a more complete result. Gertz and Schmitt propose the extended query language to deal with the query which take into account the quality feature.

```
Select [restrict] < list of attribute >
from G
where < selection condition >
using < selection feature >
with < weight feature >
```

The "where" clause applies a condition on the answer set while the "using" clause applies a feature condition on the result set.

C. *Quality-driven Integration of Heterogeneous Information Systems*

Naumann [15] has developed a system that integrates heterogeneous information systems based on data quality that identify and rank high quality plans, in order to produce results with high quality. The project looks for query plans that are correct and may produce different results while traditional optimization techniques consider plans that all produce same results.

For query processing part, the project process queries by considering the different levels of granularity for each data quality:

1) Criteria on the data source, determines the quality of the whole data source, such as timeliness and reputation.
2) Criteria on query correspondence assertion (QCA), defines the quality of specific query correspondence assertions, such as the cost of a query.
3) Criteria on user query, measures the quality of the answer delivered to a specific user query. The scores for these criteria can only be calculated at query time. An example is completeness.

However, Naumann doesn't specify how to measure these quality criteria at different levels of granularity. The project uses the Data Envelopment Analysis method [16] to rank the data sources. Therefore, user priorities are ignored at this process. Besides, data sources are discarded by subjective criteria such as reputation and understandability.

D. *Quality-Driven Query Answering for Integrated Information Systems*

Naumann [12] developed a project to integrate data from different data sources based on data quality. The project uses the global relational schema to generate a universal relation to be used for integrating autonomous data sources. Users generates queries by selecting attributes from the universal relation and may specify conditions on the selected attributes. Queries are then transformed to queries against the global relational schema. The project qualifies data sources based on several quality criteria such as objectivity, believability, reputation and others. These criteria are used to generate a quality model for query plans.

The project follows the following steps to calculate the quality of a query plan: Each source gets information quality (IQ) scores for each relevant data quality criteria. The IQ scores are, then, combined to form an IQ-vector. In order for users to determine their preferences, they are required to assign weights to the IQ-vector. Therefore, a weighting vector can be obtained.

A multi-attribute decision-making (MADM) method uses the weighting vector to rank the data sources in the universal relation. An example of MADM method is simple additive weighting (SAW).

After determining the IQ-vector for each data source, the project calculates an IQ-vector for each query plan containing the data sources. Query plans are, then, represented as trees of joins between the data sources: leaves represent data sources while inner nodes represent the joins between the data sources. By joining nodes from bottom to up, each inner node gets IQ-scores and the overall quality of the plan is the IQ-score of the root of the tree.

IV. QUALITY SYSTEM COMPONENTS

The purpose of adding quality system components to data integration systems is to improve the query answers. This can be achieved by assessing a set of quality criteria over the data sources and storing the measures in a repository to be used later during query planning phase to generate query plans that can produce answers with better quality. These quality system components are: (1) Data quality acquisition and (2) user input. These quality system components are integrated in the mediator-wrapper architecture. See green boxes in "Fig. 4."

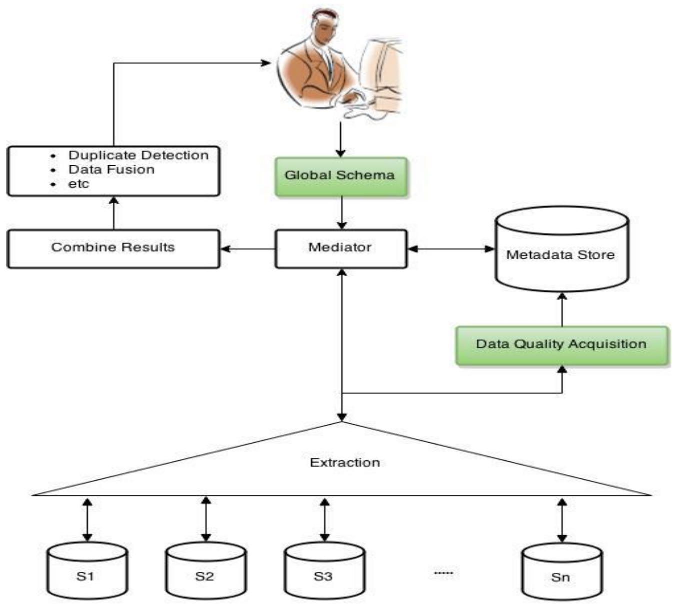


Fig. 4. Data integration system quality system components

In the following sub-sections, we present the structure and the functionality of each component.

A. *Data Quality Acquisition*

This component is responsible for extracting attributes and relations from the data sources and store them in the metadata store. It is also responsible for executing data quality queries against the data sources, receiving the results and store them in the metadata store. The metadata store used by the data quality acquisition (DQA) consists of the following relations as shown in "Fig. 5":

1) Data Sources. Stores information about data sources.
2) Tables. Stores information about each relation (table) in each data source. It has an attribute called "detectors" which is used to uniquely identify records in case no primary key exists. This attribute is used during fusion process [17].

3) Columns. Stores information about each column in each relation in a data source.
4) Global schema columns. Stores information about the global schema attributes.
5) Global schema Tables. The global schema columns belong to a global schema table. This makes it easier to add multiple tables with columns.
6) Global Schema Mappings. Defines the mapping between the attributes of the global schema and the attributes of the data sources (stored in columns relation) and the mapping functions between the attributes. Such mapping functions are multiplication, division, string concatenation, etc.

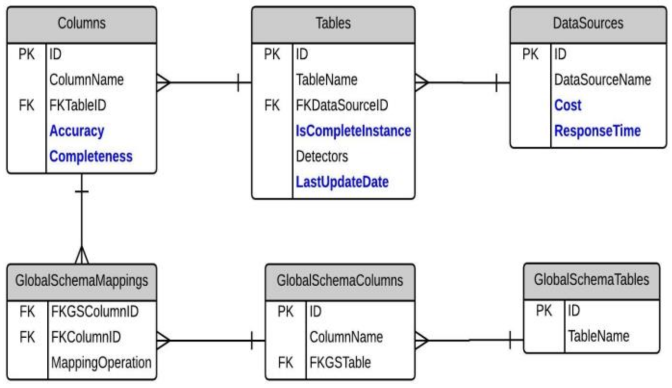


Fig. 5. Metadata structure

Data quality acquisition component can be customized without affecting the data integration system. We can change the queries used by the data quality acquisition anytime. Whenever data quality acquisition executes queries, the quality measures in the metadata store will be updated with the new values.

We have selected a set of data quality dimensions that could affect the data integration process and at the same time could be considered important to the end user to be measured by data quality acquisition. Table I illustrates these dimensions and the granularity for each dimension.

TABLE I. DATA QUALITY DIMENSIONS AND GRANULARITIES LEVELS

DQ Dimension	**Measures granularities**		
	Data source level	***Relation level***	***Attribute level***
Accuracy			✓
Completeness			✓
Cost	✓		
Response time	✓		
Timeliness		✓	

In the following sub-sections, we describe how we measure each dimension presented in table I. These quality measures may enhance the quality of the data fusion process. Data quality dimensions chosen are highlighted in blue in "Fig. 5".

1) Accuracy

Tomas C. Redman [10] present the data accuracy measurement framework ("Fig. 6") for understanding the various measurement techniques based on choices made regarding four factors: where to measure the data, which part of the data will be measured, how to measure the data and the granularity of the measures.

To apply Redman's data accuracy measurement framework in our case, we will select from the choices for each of the four factors.

- Where measurements are taken: Since we have a set of data sources given, we will measure accuracy from those data sources. (i.e. from database).
- What attributes to include: To save processing time, we measure accuracy on the data sources' attributes that correspond to global schema's attributes.
- The measurement device: Since a reference to a real world relation is almost always costly and time consuming, we will compare the value of each attribute to its domain of allowed values. Complaints and domain experts' feedback are also used to identify erred data and a correction for them which help improve accuracy measure.
- The scale on which results are reported: Attribute level.

$$\text{Attribute Accuracy} = \frac{\text{Number of fields judget correctly}}{\text{Number of fields tested}} \qquad (1)$$

2) Completeness

The Literature classifies completeness into three types: column completeness, schema completeness, and population completeness [18]. At the most abstract level, schema completeness refers to the degree to which all required information are present in a particular data set. At the data level, column completeness can be defined as the measure of the missing values for a column in a table. Each of the three types can be measured by dividing the number of incomplete items by the total number of items and subtracting from 1 [18].

$$\text{Schema/Attribute completeness} = 1 - \frac{\text{Number of incomplete items}}{\text{Total number of items}} \quad (2)$$

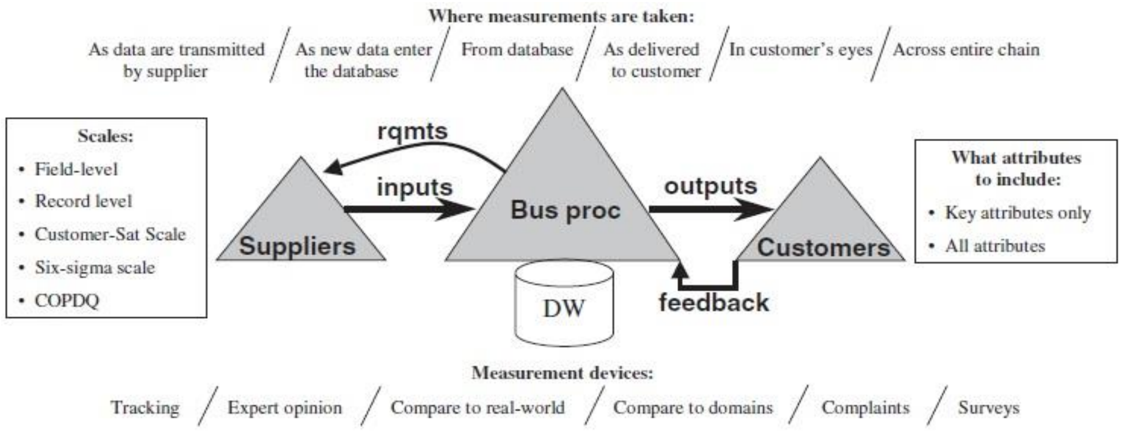


Fig. 6. The data accuracy measurement framework

The range for completeness is 0 - 1, where 0 represents the lowest score and 1 represents the high score.

In relational databases research area, completeness is often related to the meaning of "null" values. A value can be missing because it doesn't exist or because it exists, but is not known or because its exist and not identified. We apply completeness on non-primary key attributes and applicable attributes. We add a custom data quality criteria called "Complete instance relation" that can be measured at schema level. A relation is marked as complete instance if its cardinality is complete. (i.e. all the tuples are represented in the relation). This information will be given directly to the data integration system by end user through an input form.

3) Cost

It is the price for accessing specific data source. The user has to pay the money for accessing a commercial data source. We added this criteria due to the growing importance of commercial data source providers. The subscription of the user and the cost of that subscription are determined by the data source owner. We assume that the user is charged on pay-by-query basis. The cost per query is measured in US dollar.

4) Response Time

We measure the response time of a data source by using calibration techniques [19]. We send a bunch of queries to the data sources to judge their average response time for different types of queries at different times of day. The result will be stored at the metadata store to be used later during query planning phase. The response time may be high if a source is always busy or doesn't have the resources needed to answer the query. In this study, we assume that all the data sources have the capabilities to answer all queries so that the problem of source capabilities is resolved. The response time depends on several factors such as network traffic, servers workload, technical equipment such as: the hardware of the data source and database management system used by the data source.

5) Timeliness

Timeliness measure depends on the data integration system: some prefer seconds while others prefer days. To determine the timeliness, we rely on update information provided by the data source. Timeliness measurement depends on at what granularity the data source updates its data. We assume at the relation level and the data at the data sources are not archived.

B. User Input

To give users the option to specify constraints on the retrieved result, we have used the proposal of Gertz and Schmitt [14] where quality constraints can be expressed by using data quality dimensions. Thus forming a threshold of acceptance. We have added two options to the SQL dialect. The first one is cost which is the amount of money a user can pay and the second option called fusion that can be set to true or false and is used to give the user the option to fuse data from all possible data sources.

A query Q with quality constraint expressed on the mediated schema expressed in an extended SQL syntax:

```
Select A1,…..,Ak
from G
where < selection condition >
with < data quality goal >
fusion < true | false >
Cost < x$ >
Where A1.A2,…, Ai are global attributes of G
```

Selection condition: conditions used to filter the data.

Data quality goal: quality dimensions defined on the selected attribute Ai and gets a value according to table II.

TABLE II. DATA QUALITY DIMENSIONS LEVELS

Level	***Start threshold***
High	70
Meduim	50
Low	0

The values in table II are used as a threshold of acceptance. Ex: if we have a quality constraint A1.completeness is Medium, that's mean that the user wants the completeness of A1 in the answer to be 50% or higher.

The values in table II are tunable. The system administrator can change the value each level anytime.

Fusion: When set to true, this means that the user wants to fuse data from all possible data sources. When set to false, the mediator selects only one alternative that has the minimum number of data sources.

Cost: the amount in US dollar the user can pay.

V. QUALITY DRIVEN QUERY PROCESSING ALGORITHM

The requested data usually located on more than one data source. Every combination of data sources that meet the user's requirements (attributes and quality criteria) is an alternative. If a single data source can meet all user's requirement, this is an alternative. Given a query Q against the mediated schema asking for A1,…..,An attributes with or without quality requirements, We developed a quality-driven query algorithm ("Fig 7") to determine which combinations of sources can answer the query. The algorithm works as follows:

1) Since each global schema attribute is assigned a unique ID, the mediator obtains a metadata representation by joining the tables in the metadata store and using the IDs of the selected attributes to retrieve the metadata related to the selected attributes only.

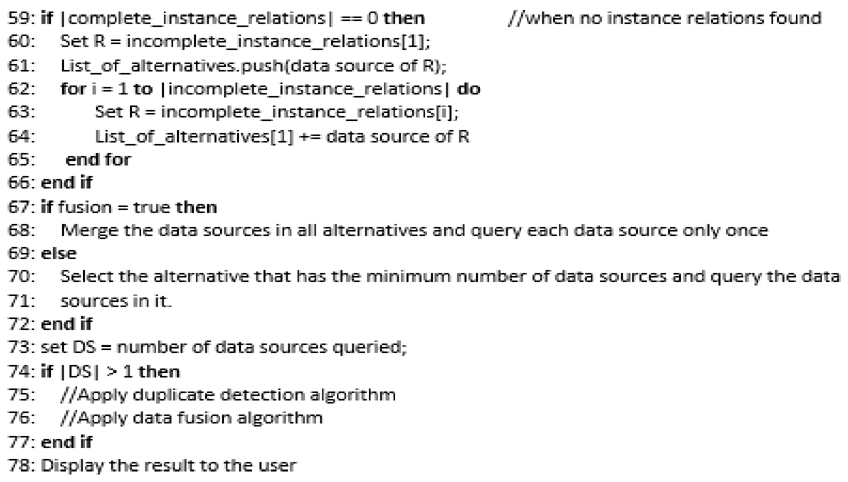
2) After determining the data sources that can answer the query, we examine the quality criteria required by the user (if exist) against the quality of these data sources. We discard the data sources that don't meet user's cost criteria. Ex: If the user can pay x$, then all data sources whose cost greater than x, will be discarded.

3) The mediator extracts the complete instance relations from the remaining data sources and examines the quality and the attributes provided by these relations against the quality and the attributes required in the user's query. This process is as following:

doi:10.15849/icit.2015.0078 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

```
Input: metadata representation
Output: list of alternatives
// means comment

1: List_of_alternatives = {}
2: List_of_warningMessages = {}
3: temporary_alternatives = {}
4: missing_attributes(key, value) = {}      // Key represents the missing or disqualified attribute while
                                            // value represents the data sources that provide that attribute.
5: for i = |complete instance relations| to 1 do
6:     Set R = complete_instance_relations[i];
7:     if R contains all required attributes && R meet all DQ criteria then
8:         List_of_alternatives.push(data source of R);
9:     else
10:        missing_attributes = {};
11:        Set A = the missing or disqualified attributes from R;
12:        for m = 1 to |A| do                  //add the missing or disqualified attributes to the collection
13:            missing_attributes.push(A[m]);
14:        end for
15:        for m = |complete_instance_relations| to 1 do
16:            Set k = complete_instance_relations[m];
17:            for z = |missing_attributes| to 1 do
18:                if k contains A[z] && k meet the DQ criteria required for A[z] (if exist) then
19:                    missing_attributes[A[z]] += the data source of k; //Add the data source of K to A[z]
20:                end if
21:            end for
22:        end for
       //Sort missing_attributes in descending order based so that the attribute that has the highest number of data sources
       //becomes first
23:        temporary_alternatives = {};
24:        if fusion = true then
25:            Set D = missing_attributes[1]; // set D = the data sources for the first element
26:            for f = |D| to 1 do
27:                temporary_alternatives.push(data source of R with D[f]);
28:            end for
29:            for f = 2 to |missing_attributes| do
30:                for w = 1 to |temporary_alternatives| do
31:                    Set D =  missing_attributes[f];  // D = data sources that provide missing_attributes[f]
32:                    temporary_alternatives[w] += D; // update each alternative with the data sources that provide
                       //missing_attributes[f] so that the alternative meets the user's needs.
33:                end for
34:            end for
35:            for w = 1 to |temporary_alternatives| do
36:                List_of_alternatives.push(temporary_alternatives[w]);
37:            end for
38:        else                          // Generate alternative for each missing attribute
39:            temporary_alternatives.push(data source of R);
40:            for f = 1 to |missing_attributes| do
41:                Set D = missing_attributes[f]; // D = data sources that provide missing_attributes[f]
42:                if |D| > 1 then
43:                    Set E = DataEnvelopmentAnalysis(D); // Apply DEA on the data sources and set the
                                                           //efficient data sources to E
44:                    for w = 1 to |E| do
45:                        temporary_alternatives[1] += E[w];
46:                    end for
47:                else
48:                    temporary_alternatives[1] += D;
49:                end if
50:            end for
51:            List_of_alternatives.push(temporary_alternatives[1]);
52:        end if
53:    end if
54: end for
55: if |List_of_alternatives| == 0 then //No data source found that can match the required DQ criteria
56:     List_of_warningMessages.push("Sorry, no data sources found match the DQ criteria required for attribute x");
57:     List_of_alternatives.push(data source of R); // we must provide answer. Even partial answer
58: end if
```



```
if |complete_instance_relations| == 0 then            //when no instance relations found
    Set R = incomplete_instance_relations[1];
    List_of_alternatives.push(data source of R);
    for i = 1 to |incomplete_instance_relations| do
        Set R = incomplete_instance_relations[i];
        List_of_alternatives[1] += data source of R
    end for
end if
if fusion = true then
    Merge the data sources in all alternatives and query each data source only once
else
    Select the alternative that has the minimum number of data sources and query the data
    sources in it.
end if
set DS = number of data sources queried;
if |DS| > 1 then
    //Apply duplicate detection algorithm
    //Apply data fusion algorithm
end if
Display the result to the user
```

Fig. 7. Quality driven query algorithm

a) If a relation R in data source S doesn't provide at least one attribute then the mediator examines other data sources for that missing attribute.

b) If a relation R in data source S doesn't meet the quality criteria required for at least one attribute then the mediator examines other data sources for that disqualified attribute.

c) If a relation R in data source S provides all required attributes and meets all the quality criteria required in Q, then no other data sources are needed and the mediator adds the data source of R to the list of alternatives.

d) The mediator applies the above three steps for the remaining complete instance relations.

4) If no complete instance relation is found in step 3, then all data sources will be used to form an alternative.

5) The following steps illustrate how the mediator finds other data sources that provide the missing attributes or the disqualified attributes:

a) The mediator checks the metadata representation for data sources that provide the required attributes.

b) If no relations found in step 5-a, that means that one of the required quality criteria can't be factory. So, a warning message will be displayed to the user regarding that quality criteria unless an alternative is found. However, the mediator must provide an answer. So, we add the data source of R (R is defined in step 3) to the list of alternatives.

c) If relations are found in step 5-a, then the mediator checks if fusion option in Q is set to true or false.

- If fusion is set to true, the mediator sorts the missing attributes in a descending order based on the number of data sources found for each attribute. The reason for that is to make sure that for each data source S_1, S_2, …, Sn that provide the first missing attribute, the mediator generates alternatives that consists of the data source of R (R is defined in step 3) and Si. Then the mediator iterates on the remaining attributes and updates the alternatives with data sources that provide each missing attribute. Finally, the alternatives are then added to the list of alternatives.
- If fusion is set to false, the mediator generates only one alternative for all required attributes. To find that alternative, for each missing attribute A_1,A_2,…, Ai, we apply Data Envelopment Analysis (DEA) method on the data sources that provide Ai and any efficient source will be added with the data source of R to the list of alternatives. If no efficient data sources are found, the highest inefficient data source will be added with the data source of R to the list of alternatives. The quality measures, which are already stored in the metadata store, will be used in DEA. These quality measures are: Completeness, Accuracy, Cost and Response time.

6) After generating all alternatives, the mediator checks if fusion option in Q is set to true or false.

- If fusion is set to true, the mediator merges the data sources in all alternatives and query each data source only once.
- If fusion is set to false, the mediator selects the alternative that has the minimum number of data sources and query the data sources in it.
- If the number of queried data sources is greater than one, duplicate detection and data fusion algorithms will be run on the result respectively.
- The result is then displayed to the user.

VI. EVALUATION AND VALIDATION

In this section, we validate that our approach really reduces the number of data sources needed to answer a given query.

Given the Student schema in data sources S1, S2, S3, S4, S5, S6 and S7. The mediated schema is shown below

G: Student (FirstName, LastName, Gender, Birthdate, Mail, Nationality, Address, Phone)

S1.Student (StudentID, FirstName, LastName, Gender, Birthdate, Address)

S2.Students (SID, Name, Sex, Birthdate, E-mail, Nationality)

S3.Student (ID, FullName, Email_Address, Nationality, Phone)

S4.Student (Id, FName, LName, Gender, Mail, Nationality, Address, Phone)

S5.Student (Student_id, Student_Name, Gender, Nationality, Birthdate)

S6.Student (Student_ID, Name, Gender, Mail, Phone)

S7.Student (S_ID, FName, LName, Sex, Address, Birthdate)

The data sources measures in the metadata store are shown in table III.

TABLE III. DATA SOURCES MEASURES IN THE METADATA STORE

Data Source ID	**Properties**		
	Data Source Name	***Response Time***	***Cost***
1	S1	92 sec	10$
2	S2	160 sec	5$
3	S3	130 sec	3$
4	S4	280 sec	0$
5	S5	500 sec	0$
6	S6	350 sec	7$
7	S7	300 sec	0$

Now, Consider the following query Q1:

Select FirstName, LastName, Gender, Birthdate
From G
Where Gender =”Male”
With Birthdate.completeness is high
fusion = true
Cost = 0

The interpretation of the above query is that the user wants First Name, Last Name, Gender and Birthdate of all male students where completeness of birthdate is high (i.e. completeness measure starts from 70 as indicated in table II) and obtain the result from the free data sources only.

The first step to process the above query is by obtaining metadata representation by joining the tables in the metadata store (see “Fig. 5”) and filtering the rows by the IDs of the selected attributes to retrieve the metadata related to the selected attributes only.

Second, we discard the data sources that don’t meet the cost criteria specified in Q. Therefore, S1, S2, S3 and S6 are discarded. This yields the metadata representation shown in table IV.

TABLE IV. METDATA REPRESENTATION

Properties					
Column Name	***Table***	***Accuracy***	***Completeness***	***Table Name***	***Complete instance***
FName	S4	99	100	Student	Yes
LName	S4	97	100	Student	Yes
Gender	S4	100	100	Student	Yes
Student_ Name	S5	88	95	Student	Yes
Gender	S5	100	100	Student	Yes
Birthdate	S5	80	84	Student	Yes
FName	S7	100	100	Student	Yes
LName	S7	100	100	Student	Yes
Sex	S7	100	100	Student	Yes
Birthdate	S7	80	90	Student	Yes

Third, the mediator extracts the complete instance relations from the remaining data sources. From table IV, complete instance relations are S4, S5 and S7. The mediator starts with S4 and finds that S4 does provide all required attributes except Birthdate, so the mediator will look for other data sources that could provide attribute “Birthdate”. After scanning the metadata representation (table IV), the mediator finds S5 and S7. The mediator then checks the fusion option in Q. Since fusion option is set to true, the mediator generates an alternative for each data source. So, the list of alternatives = {{S4, S5}, {S4, S7}}.

Next, the mediator examines S5 and finds that S5 does provide all required attributes (S5.Student.Student_Name contains FirstName concatenated with LastName). So no other data sources are needed. Therefore, S5 itself is an alternative and the mediator will add it to the list of alternatives. Therefore, the list of alternatives = {{S4, S5}, {S4, S7}, {S5}}.

Next, the mediator examines S7 and found that S7 does provide the required attributes. So, no other data sources are needed. Therefore, S7 itself is an alternative and the mediator adds it to the list of alternatives. Therefore, the list of alternatives = {{S4, S5}, {S4, S7}, {S5}, {S7}}.

Now, given a set of alternatives, the mediator determines which alternatives to choose as follows:

- The mediator checks again fusion option in Q. Since the fusion is set to true, the mediator merges the data sources in all alternatives and query each data source only once. Therefore, the final query plan = {S4, S5, S7}.
- After retrieving the result from each data source, the mediator unions the results and applies duplicate detection algorithm to find the tuples that refer to the same real world entity.

- After determining the duplicate records, a data fusion algorithm is needed to fuse attribute's values that refer to the same real world entity (resolve inconsistency at value level).
- Any further processing on the result can be applied.
- Finally, the result is displayed to the user.

IF we assume the complete instance property for S4 is "No", we will find that the final list of alternatives {{S5}, {S7}} and the final query plan as {S5, S7}. S4 is omitted because it doesn't provide a complete result. This reduces the time needed to answer a query by avoiding access to S4.

If we modify fusion option and set it to false and repeat the above steps, we will find that the metadata representation is still the same (table IV). S4 does provide all required attributes except Birthdate. The mediator selects S5 and S7 to provide attribute Birthdate. Since fusion is set to false, the mediator tries to choose the best source between S5 and S7 to retrieve attribute birthdate from. The mediator achieves this by applying DEA on S5 and S7. The quality scores in table V are used in DEA.

TABLE V. QUALITY SCORES

Data Source	**Quality Criteria**			
	Accuracy	***Completeness***	***Response Time***	***Cost***
S5	80	84	500 sec	0$
S7	80	90	300 sec	0$

The computed efficiency for S5 is 0.7939 while the computed efficiency for S7 equals 1. Therefore, the mediator adds S7 along with S4 as an alternative to the list of alternatives. The list of alternatives = {{S4, S7}}.

Next, the mediator examines S5 and S7 respectively and finds both provide all required attributes. Therefore, S5 and S7 themselves are alternatives and the mediator adds them to the list of alternatives. Therefore, the list of alternatives = {{S4, S7}, {S5}, {S7}}.

Now, given a set of alternatives, the mediator determines which alternatives to choose as the following:

- Since the fusion is set to false, the mediator chooses the alternative that has the minimum number of data sources and query the data sources in it. In this case, the mediator can choose either S5 or S7. Therefore, the final query plan = {S7}.
- Since the number of data sources queried equals one, neither duplicate detection algorithms nor data fusion algorithms are needed unless the data source allows duplicate.
- Any further processing on the result can be applied.
- Finally, the result is displayed to the user.

VII. CONCLUSION AND FUTURE WORK

Data integration systems may produce query results that not only suffer the lack of quality but also take a long time to arrive. The results can be incomplete, inaccurate or outdated and so on. In this paper, we have pointed out the importance of data quality in integrating autonomous data sources. The main contribution of this paper is an efficient method aimed at selecting a few possible data sources to provide more quality oriented result to the user. We added quality system components to integrate data quality dimensions in a data integration environment for structured data sources only. With the help of these criteria, we developed a quality driven query execution algorithm to generate high quality plan that meets user's requirements. Further research will extend the approach to be applied on different types of data sources such as semi-structured and unstructured data sources.

REFERENCES

[1] M. Lenzerini, "Data integration: a theoretical perspective," in Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database system, Madison, Wisconsin, June 03-05, 2002.

[2] A. Y. Halevy, "Answering queries using views: A survey," The VLDB Journal — The International Journal on Very Large Data Bases, vol. 10, no. 4, pp. 270-294, December 2001.

[3] A. Doan, A. Halevy and Z. Ives, Principles of Data Integration, San Francisco, CA: Morgan Kaufmann Publishers Inc., 2012.

[4] G. Wiederhold, "Mediators in the Architecture of Future Information Systems," IEEE Computer, vol. 25, no. 3, pp. 38-49, March 1992.

[5] Y. Wand and R. Y. Wang, "Anchoring data quality dimensions in ontological foundations," Communications of the ACM, vol. 39, no. 11, pp. 86-95, November 1996.

[6] M. Ge and M. Helfert, "A Review of Information Quality Research - Develop a Research Agenda," in Proceedings of the 12th International Conference on Information Quality (ICIQ 07), MIT, Massachusetts, USA, November 9-11, 2007.

[7] J. M. Juran, The Quality Control Handbook, 3rd ed., New York: McGraw-Hill, 1974.

[8] G. Kumar Tayi and D. P. Ballou, Examining data quality, Communications of the ACM,, v.41 n.2, p.54-57, Feb. 1998.

[9] R. Y. Wang and D. M. Strong, "Beyond accuracy: what data quality means to data consumers," Journal of Management Information Systems, vol. 12, no. 4, pp. 5-33, March 1996.

[10] T. C. Redman, "Measuring Data Accuracy A Framework and Review," in Information Quality, R. Y. Wang, E. M. Pierce, S. E. Madnick and C. W. Fisher, Eds., Armonk, NY, M.E. Sharpe, 2005, pp. 21-36.

[11] C. Batini and M. Scannapieco, Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications), Secaucus, NJ: Springer-Verlag New York, Inc, 2006.

[12] F. Naumann, Quality-Driven Query Answering for Integrated Information Systems, Springer-Verlag Berlin, Heidelberg, 2002.

[13] M. Scannapieco, A. Virgillito, C. Marchetti, M. Mecella and R. Baldoni, "The daquincis architecture: a platform for exchanging and improving data quality in cooperative information systems," Information Systems, vol. 29, no. 7, pp. 551 - 582, October 2004.

[14] M. Gertz and I. Schmitt, "Data integration techniques based on data quality aspects," in 3rd National Workshop on Federal Databases, Magdeburg, Germany, 1998.

[15] F. Naumann, U. Leser and J. C. Freyta, "Quality-driven integration of heterogenous information systems," in 25th proceeding of the International Conference on Very Large Databases (VLDB) , p.447-458, Edinburgh, Scotland, September 07-10, 1999.

[16] A. Charnes, W. W. Cooper and L. Rhodes, "Measuring the efficiency of decision making units," European Journal of Operational Research, vol. 2, no. 6, pp. 429-444, November 1978.

[17] A. Z. El Qutaany, A. H. El Bastawissy and O. Hegazy, "A Technique for Mutual Inconsistencies Detection and Resolution in Virtual Data Integration Environment," in Informatics and Systems (INFOS), 2010 The 7th International Conference on, 2010.

[18] L. L. Pipino, Y. W. Lee and R. Y. Wang, "Data quality assessment," Communications of the ACM, vol. 45, no. 4, pp. 211-218, April 2002.

[19] M. Spiliopoulou, "A calibration mechanism identifying the optimization technique of a multidatabase participant," in Proc. of the Conf. on Parallel and Distributed Computing Systems (PDCS), Dijon, France, September 1996.