A COMPLETENESS-AWARE DATA QUALITY PROCESSING APPROACH FOR WEB QUERIES

Sandra de F. Mendes Sampaio
School of Computer Science, University of Manchester, Oxford Road, Manchester, U.K.

Pedro R. Falcone Sampaio
Manchester Business School, University of Manchester, Manchester, U.K.

Keywords: Data Quality, Internet Query Systems, Completeness, Query Processing.

Abstract: Internet Query Systems (IQS) are information systems used to query the World Wide Web by finding data sources relevant to a given query and retrieving data from the identified data sources. They differ from traditional database management systems in that data to be processed need to be found by a search engine, fetched from remote data sources and processed taking into account issues such as the unpredictability of access and transfer rates, infinite streams of data, and the ability to produce partial results. Despite the powerful query functionality provided by internet query systems when compared to traditional search engines, their uptake has been slow partly due to the difficulty of assessing and filtering low quality data resulting from internet queries. In this paper we investigate how an internet query system can be extended to support data quality aware query processing. In particular, we illustrate the metadata support, XML-based data quality measurement method, algebraic query processing operators, and query plan structures of a query processing framework aimed at helping users to identify, assess, and filter out data regarded as of low completeness data quality for the intended use.

1 INTRODUCTION

There has been an exponential growth in the availability of data on the web and in the usage of systems and tools for querying and retrieving web data. Despite the considerable advances in search engines and other internet technologies for dynamically combining, integrating and collating web data, supporting a DBMS-like data management approach across multiple web data sources is still an elusive goal. To buck this trend, internet query systems − IQS (Naughton, J., DeWitt, D., Maier, D., et al, 2001) are being developed to enable DBMS-like query processing and data management over multiple web data sources, shielding the user from complexities such as information heterogeneity, unpredictability of data source response rates, and distributed query execution.

The comprehensive query processing approach supported by IQS allows users to query a global information system without being aware of the sites structure, query languages, and semantics of the data repositories that store the relevant data for a given query (Naughton, J., DeWitt, D., Maier, D., et al, 2001). Despite the significant amount of work in the development of the data integration and distributed query processing capabilities, internet query systems still suffer from inadequate data quality control mechanisms to address the management of quality of the data retrieved and processed by the IQS. Typical examples of data quality issues (Olson, J., 2003) that need to be addressed when supporting quality aware query processing over multiple web data sources are:

- Accuracy of data: Data can have errors or inconsistencies in its representation. For example, the data values “St. Louis” and “Saint Louis” may not be matched in a join operation between data sources despite referring to the same address instance, due to the different representation formats.
- Completeness of data: a data source is regarded as complete if all information requirements are modelled and stored in the database. For instance, data sources fed by online forms with poor data quality checks and optional fields often give rise to

de F. Mendes Sampaio S. and R. Falcone Sampaio P. (2008).
A COMPLETENESS-AWARE DATA QUALITY PROCESSING APPROACH FOR WEB QUERIES.
In *Proceedings of the Third International Conference on Software and Data Technologies - ISDM/ABF*, pages 234-239
DOI: 10.5220/0001894802340239

data with low completeness due to several attributes with null values.

• Timeliness of data: Data can be of poor quality when it is not timely for the intended use.

Internet query systems support the mediator-based approach to quality management (Wiederhold, G., 1992). The mediator-based approach is applicable in situations where users need to formulate complex queries encompassing multiple web data sources for which there is no control over the data available in a data source and the infrastructure supporting data source site query processing (e.g., querying e-Science data sources [http://www.rcuk.ac.uk/escience]). In the mediator-based approach, the speed of the internet limits transmission of relevant data, and users cannot reconcile and cleanse all necessary data items prior to query formulation as in data warehouse-based quality-based integration approaches (Helfert, M., and E. von, Maur, 2001), therefore needing dynamic strategies for managing accuracy, completeness, and timeliness data quality issues.

In this paper we investigate how an internet query system can be extended to support a dynamic data quality aware query processing framework. In particular, we illustrate the completeness assessment method, metadata support, algebraic query processing operators, and query plan structures of a query processing framework aimed at helping users to identify, assess, and filter out data regarded as of low completeness quality for the intended use. The remainder of the paper is structured as follows. Section 2 discusses key contributions of the proposed approach. Section 3 presents how completeness measures can be associated with XML data. Section 4 presents the method used to annotate XML data with completeness information. Section 5 describes the quality aware algebraic query processing framework. Section 6 provides an example illustrating how completeness is assessed during query processing. Section 7 describes related work. Section 8 summarizes the work.

2 DISCUSSION

The main contribution of this paper is to demonstrate how existing ideas in the arena of query processing and quality of information can be combined and applied in the context of Internet data processing, to analyse, evaluate and possibly filter data of unacceptable Completeness quality. The ideas being reused are: query engine extension, and data annotation with quality information. The approach taken in this work suggests Internet data to be annotated with quality information prior to query processing, to allow the quality information to be taken into consideration during query optimisation and execution. The data annotation can be done automatically by the mediator system, or by data providers, if these are cooperative data sources.

In addition to annotating data with quality information, the query engine of the mediator system requires extension to enable quality information to be taken as input, processed along its corresponding data, and have an impact on the produced results, which should reflect what the user requested in his/her query. The proposed framework is tested over a simple, extensible, and robust Internet Query System (Naughton, J., DeWitt, D., Maier, D., et al, 2001), which receives, as input, Internet data in XML format and generates, as output, XML data representing results from submitted user queries. The query engine is implemented in a Database Management System fashion, i.e., as a set of algebraic operators designed to work together to process XML data. More specifically, this work shows how the query engine algebra can be extended with an operator encapsulating capabilities to deal with information about the Completeness quality of the data being processed. The approach of query engine extension has a number of shortcomings, such as the ones described as follows: It may increase the complexity of design and implementation of the engine. Therefore, modularity and encapsulation are important features to be taken into consideration when designing a quality-aware query engine for any system, always enforcing the idea that data quality related functionalities must be all encapsulated within "data quality" operators; it may be intrusive in the point of view of traditional query processing, as the addition of new operators/functions into a pre-existing engine can demand the creation of new optimisation rules and heuristics, as well as modification of pre-existing operators. We believe that a careful query engine design and implementation can avoid problems of intrusiveness, as it has been shown in previous work describing extension of engines to deal with parallelism in query execution (Graefe, G., 1996); it may be inflexible or difficult in allowing the user to build and incorporate into the system his own definition of quality based on the task he/she has at hand. Flexibility in allowing user input can be achieved by providing user interfaces to take user input or feedback on quality of data.

3 MEASURING COMPLETENESS

Completeness is a context-dependent data quality dimension that refers to "the extent to which data are of sufficient breadth, depth and scope for the task at hand" (Wang, R., and S. E., Madnick, 1989). In the context of a database model, two types of completeness dimensions are considered: *model completeness* and *data completeness*. Model completeness refers to the measure of how appropriate the schema of the database is for a particular application. Data completeness refers to the measurable errors of omission observed between the database and its schema, checking, for example, if a database contains all entities/attributes specified in the schema.

Completeness issues arising in database applications may have several causes, for example, discrepancies between the intent for information querying and the collected data, partial capture of data semantics during data modelling, and the loss of data resulting from data exchange. Potential approaches to address completeness issues include removing entities with missing values from the database; replacing missing values with default values, and completing missing values with data from other sources. Irrespective of the approach taken to deal with poor data completeness, it is crucial that database users formulating queries across multiple data sources are able to judge if a particular query result is "fit" for its purpose, by measuring the level of completeness of the result.

4 ANNOTATING XML DATA FOR COMPLETENESS

To enable quality aware query processing, data should be annotated with quality information (Wang, R. Y., Reddy, M. P., and Kon, H. B., 1995). Annotations describing simple data completeness information can be done automatically by the mediator system, as streams of data from remote data sources are input. The information should specify the number of tag elements and element values missing from an XML document, relative to the expected numbers for the document to be considered complete. This information can be obtained by simply counting the numbers, while parsing and analysing the structure of the document against its schema description or DTD.

```
<carDealerInformation>
   <dealer id="id001">
      <name>Audi Dealers</name>
         <car><model>A6 Avant</model>
              <price>26000</price></car>
   </dealer>
   <dealer id="id002">
      <name>Fiat Dealers</name>
      <car><model>Cinquecento</model>
           <price>8000</price></car>
      <car><model>Idea</model></car>
      <car><model>Multipla</model>
           <price></price></car>
   </dealer>
   <dealer id="id003">
      <name>Renauld Dealers</name>
   </dealer>
   <dataQuality><completeness>
      <numberElements>22</numberElements>
      <missingElements>
         <numberMissingElem>1</numberMissingElem>
         <elem><name>price</name>
         <number>1</number></elem>
      </missingElements>
      <numberValues>14</numberValues>
      <missingValues>
         <numberMissingVal>2</numberMissingVal>
         <elem><name>price</name>
         <number>2</number></elem>
      </missingValues>
   </completeness></dataQuality>
</carDealerInformation>
```

Figure 4.1: Example of XML document with annotations.

The annotated numbers for missing tag elements and element values, as well as expected numbers, represent quality factors that will be taken into account during query execution. These quality factors are added into an XML document as (sub) elements and (sub) element values associated with other elements specified in the schema or DTD of the original XML document. An example is illustrated in Figure 4.1, showing an XML document describing information about car dealers. Note that information to be used during query processing to calculate the completeness of `carDealerInformation` is attached to the original document, following an initial parsing of the document and its schema. In this example, both the price element and its value for car `Idea` are missing. It is also missing the price value for car `Multipla`.

5 QUALITY AWARE QUERY PROCESSING

Figure 5.1 illustrates the quality aware query processing framework proposed in this work, which can be implemented as an extension to Internet Query Systems.

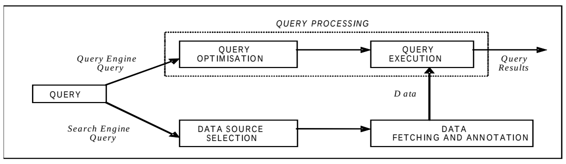

Figure 5.1: Query Processing and Data Search in an IQS.

In the case of Niagara, an input XML-based query expression is initially transformed into two sub-queries, a search engine query, and a query engine query. The first is used within the search engine to select data sources that are relevant to the query. Once data sources are selected, the process of fetching data takes place, and streams of data start flowing from the data sources to the site of the Internet Query System for data annotation and query execution. The second sub-query is optimised and ultimately mapped into a quality aware query execution plan that contains a special-purpose operator addressing the annotated completeness information.

A description of the query engine algebra used to execute the query plan is detailed in Table 5.1. The Completeness Algebra is an XML algebra extended with an operator that encapsulates the capability of measuring completeness quality of XML data based on completeness factors annotated on the data. This operator is called Completeness and it encapsulates functions for measuring, inserting, and propagating completeness information in XML data, provided the data has completeness factors associated with it.

Table 5.1: Completeness algebra.

Logical Operators	Description
Scan(inputData)	Builds a data structure for each data unit and passes each structure to the next operator.
Select(input,pred)	Applies a predicate (pred) over the input and either discards or retains the input depending on whether pred evaluates to false or true.
Project(input,listElem)	Discards from the input all the elements that are not specified in listElem.
Join(inputLeft, inputRight,pred)	Concatenates both inputs, retaining all their elements, applies a predicate over the result.
Completeness(input)	Updates completeness info at every step of execution, measures the final completeness score, attaches the measure to query results, and displays the results to the user.

6 QUERY EXAMPLE

Consider the XML document described in Figure 4.1, and the example XML-QL query described in Figure 6.1, which retrieves the `model` and `price` of each car offered by Fiat Dealers. Following the input to the Niagara System, the Query Optimiser generates the query plan from the query expression in Figure 6.1, illustrated in Figure 6.2. Note that, following each operator, there is a Completeness operator updating the completeness information at each step of query execution. The query results are shown in Figure 6.3. Note that only the model and price for each `Fiat Dealers`' cars appear in the results, as specified in the query. Therefore, the measures for model completeness and data completeness, performed by the Completeness operator at the root of the plan, are calculated considering only these two elements. The formulas used to calculate MC and DC are illustrated in Figure 6.4. They were derived from the ideas discussed in (Pipino, L.L., Lee, Y.W. and Wang, R.Y., 2002), which suggest that a metric to calculate the completeness score for a relational database can be formulated using simple ratio. In the simple ratio method, if the number of relations and attributes that are missing from the database is divided by the total number of relations and attributes defined in the database schema, and the result of that is subtracted by 1, then what is obtained is a number in a continuous scale between 0 and 1, that represents the model completeness score for the database relative to its schema. To measure data completeness of a relational database the same method applies, but the ration in this case should be between the number of missing attribute instances and the expected number of attribute instances. Within the continuous scale, 1 represents the highest completeness measure and is appropriate for data complying with the most strict completeness threshold, and 0 represents the lowest model completeness measure, appropriate for data that are unacceptable from the model completeness perspective.In Figure 6.2, there is a sequence of 2 pairs (Scan ,Completeness) operators, omitted for space limitations. Each of the Scan operators in the sequence unnests a level of nested elements, by attaching a copy of each unnested element (and its sub-elements) to the input tuple. For example, the first Scan unnests the `<dealer>` elements, which are sub-elements to `<carDealerInformation>`. The second Scan unnests the `<name>` and `<car>` elements, which are sub-elements to `<dealer>`. The Construct operator is the physical counterpart to the Project operator described in Table 5.1. It projects elements and builds a structure to hold query results.

Table 6.1 illustrates the functionality of the Completeness operator at each execution step of the example query. The first Completeness operator (the one following the first Scan operator) receives the original Completeness information from the data sources. Then it creates a copy of the information, attaches it to the extended layer of elements unnested by Scan, and updates the information. The updated information relates to the unnested elements.

```
WHERE <carDealerInformation>
         <dealer>
            <name>$v14</>
            <car><model>$v16</>
                  <price>$v17</>
            </></></>
IN "*" conform_to "file: completeness.dtd",$v14 =
"Fiat Dealers"
CONSTRUCT
<result>
    <model>$v16</>
    <price>$v17</> </>
```

Figure 6.1: Example query.

The second and third Completeness operators behave in a similar way, copying and updating the input Completeness information according to the changes made by the previous operator. The last Completeness operator follows Construct. It updates the number of elements and values projected by Construct, and, also, calculates the measures of MC and DC for the input document relative to the example query.

```
Completeness
     |
  Construct
     |
Completeness
     |
   Select
     |
     …
     |
Completeness
     |
    Scan
```

Figure 6.2: Query plan for example query.

```
<result>
    <model>Cinquecento</>
    <price>8000</>
</>
<result>
    <model>Multipla</>
    <price></>
</>
<modelCompleteness>0.66</>
<dataCompleteness>0.50</>
```

Figure 6.3: Example query results.

```
MC = 1–[(num of missing elements) / (num of
elements)]

DC = 1–[(num of missing values) / (num of values)]
```

Figure 6.4: Formulas used within the Completeness operator, to calculate model completeness (MC) and data completeness (DC).

Table 6.1: Behaviour of Completeness Operators.

Information updated by 1st Completeness Operator	
numberElements	21
numberMissingElements	1
nameMissingElem	<price>
numberTimes	1
numberValues	14
numberMissingValues	2
nameElemMissingValue	<price>
numberTimes	2
Information updated by Last Completeness Operator	
numberElements	6
numberMissingElem	2
nameMissingElem	<price>
numberTimes	1
nameMissingElem	<model>
numberTimes	1
numberValues	6
numberMissingValues	3
nameElemMissingValue	<price>
numberTimes	2
nameElemMissingValue	<model>
numberTimes	1
MC	0.66
DC	0.50

7 RELATED WORK

In (Mecella, M., Scannapieco, M., Virgillito, A., Baldoni, R., Catarci, T., and Batini, C., 2003) an approach for data quality management in Cooperative Information Systems is described. The architecture has as its main component a Data Quality Broker, which performs data requests on all cooperating systems on behalf of a requesting system. The system, however, does not adopt an algebraic query processing framework and is not built on top of a mainstream IQS. In (Naumann, F., Lesser, U., and Freytag, J., 1999), data quality is incorporated into schema integration by answering a global query using only queries that are classified as high quality and executable by a subset of the data sources. This is done by assigning quality scores to queries based on previous knowledge about the data to be queried, considering quality dimensions such as completeness, timeliness and accuracy. The described approach, however, does not use XML as the canonical data model and does not address physical algebraic query plan implementation issues.

8 CONCLUSIONS AND FUTURE WORK

With the ubiquitous growth, availability, and usage of data on the web, addressing data quality requirements in connection with web queries is emerging as a key priority for database research (Gertz, M., Ozsu, T., Saake, G., and Sattler, K., 2003). There are two established approaches for addressing data quality issues relating to web data: data warehouse-based, where relevant data is reconciled, cleansed and warehoused prior to querying; and mediator-based where quality metrics and thresholds relating to cooperative web data sources are evaluated "on the fly" at query processing and execution time. In this paper we illustrate the query processing extensions being engineered into the Niagara internet query system to support mediator-based quality aware query processing for the completeness data quality dimension. We are also addressing the timeliness dimension (Sampaio, S. F. M., Dong, C., and Sampaio, P. R. F, 2005) and extending SQL with data quality constructs to express data quality requirements (Dong, C., Sampaio, S. F. M., and Sampaio, P. R. F., 2006). The data quality aware query processing extensions encompass metadata support, an XML-based data quality measurement method, algebraic query processing operators, and query plan structures of a query processing framework aimed at helping users to identify, assess, and filter out data regarded as of low completeness data quality for the intended use. As future plans we intend to incorporate accuracy data quality support into the framework and benchmark the quality/cost query optimiser in connection with a health care application (Dong, C., Sampaio, S. F. M., and Sampaio, P. R. F., 2005).

REFERENCES

Naughton, J., DeWitt, D., Maier, D., et al, 2001. The Niagara Internet Query System. IEEE Data Eng. Bull. 24(2), 27-33.

Olson, J., 2003. *Data Quality: the Accuracy Dimension*, Morgan Kauffmann. 1st edition.

http://www.rcuk.ac.uk/escience. The UK e-Science Programme.

Wiederhold, G., 1992. Mediators in the Architecture of Future Information Systems. IEEE Computer 25(3).

Helfert, M., and E. von, Maur, 2001. A Strategy for Managing Data Quality in Data Warehouse Systems. In Proc. of Information Quality, 62-76.

Wang, R., and S. E., Madnick, 1989. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. Proc. of ICDE, 46-55.

Wang, R. Y., Reddy, M. P., and Kon, H. B., 1995. Toward Quality Data: An Attribute-Based Approach. Decision Support Systems, 13(3-4), 349-372.

Sampaio, S. F. M., Dong, C., and Sampaio, P. R. F, 2005. Incorporating the Timeliness Quality Dimension in Internet Query Systems. WISE 2005 Workshops, LNCS 3807, 53-62.

Dong, C., Sampaio, S. F. M., and Sampaio, P. R. F., 2006. Expressing and Processing Timeliness Quality Aware Queries: The DQ2L Approach. International Workshop on Quality of Information Systems, ER 2006 Workshops, LNCS 4231, 382-392.

Naumann, F., Lesser, U., and Freytag, J., 1999. Quality-driven Integration of Heterogeneous Information Systems. In Proc. of the 25th VLDB, 447-458.

Mecella, M., Scannapieco, Et. Al.. The DaQuinCIS Broker: Querying Data and Their Quality in Cooperative Information Systems. LNCS 2800.

Dong, C., Sampaio, S. F. M., and Sampaio, P. R. F., 2005. Building a Data Quality Aware Internet Query System for Health Care Applications. In Proceedings of IRMA Conference - Databases Track, San Diego, USA.

Graefe, G., 1996. Iterators, Schedulers, and Distributed-memory Parallelism. In Software, Practice and Experience, 26(4), 427-452.

Gertz, M., Ozsu, T., Saake, G., and Sattler, K., 2003. Data Quality on the Web. Germany, Dagstuhl Seminar.

Pipino, L.L., Lee, Y.W. and Wang, R.Y., 2002. Data Quality Assessment. CACM(45),4 (virtual extension).