Peer-to-Peer Data Quality Improvement in the DaQuinCIS System

Diego Milano, Monica Scannapieco, and Tiziana Catarci
Università di Roma “La Sapienza”
Dipartimento di Informatica e Sistemistica
Via Salaria 113, 00198 Rome, Italy
(milano, monscan, catarci)@dis.uniroma1.it

ABSTRACT: Data quality improvement is becoming an increasingly important issue. In contexts where data are replicated among different sources, data quality improvement is possible through extensive data comparisons; whereas copies of same data are different because of data errors, comparisons help to reconcile such copies. Record matching algorithms can support the task of linking different copies of the same data in order to engage reconciliation activities; for instance, a periodical running of record matching algorithms can be performed in order to reconcile copies with different quality. Nevertheless, the extensive running of such algorithms is typically performed in fixed instants. This allows for periods in which the quality of data can deteriorate, while no quality improvement action is performed on data. In this paper, we describe the DaQuinCIS (Data Quality in Cooperative Information Systems) approach for data quality improvement in contexts where data are replicated among heterogeneous and distributed sources. The DaQuinCIS strategy complements a periodical record matching activity with an “on-line” quality improvement, performed at query processing time. We experimentally show the feasibility and effectiveness of our approach by applying it to real databases; we also quantitatively evaluate the efficiency of our system.

Categories and Subject Descriptors
E.2 [Data Storage Representations]: H.2 [Database Management];

General Terms
Data Quality Management, Query Processing

Keywords: data quality, instance reconciliation, P2P systems

Received 27 Sep. 2004; Revised and accepted 13 Mar. 2005

1 Introduction

Data quality is more than simply data accuracy. It is a complex concept defined by other dimensions such as currency, completeness, consistency; such dimensions often depend from the context where data are used and also from specific users within a given context [26]. It follows that it’s not easy to obtain good quality data. Despite this complexity, the crucial role of data requires us to face and solve data quality problems.

Addressing data quality problems is particularly important in contexts characterized by extensive data replication, such as in Cooperative Information Systems (CISs). CISs are defined as all distributed and heterogeneous information systems that cooperate by sharing information, constraints, and goals [18]. CISs include data integration systems as well as systems that share common information while not explicitly integrating data. We argue that quality of data is a necessary requirement for a CIS, i.e. CISs need data quality. First, a system in the CIS will not easily exchange data with another system without a knowledge on their quality, and cooperation becomes difficult without data exchanges. Second, when poor quality data are exchanged, there is a progressive deterioration of data stored in the whole CIS. Indeed, uncontrolled exchanges of low quality data cause a diffusion of such data throughout the system, thus lowering the quality of each data asset in the CIS. Third, when a CIS is a data integration system, data integration itself cannot be performed if data quality problems are not fixed. As an example, results of queries executed over local sources must be reconciled and merged, and quality problems resulting from a comparison of

results need to be solved in order to provide the data integration system with the required information [4]. On the other hand, data replication in the CIS can be exploited for improving data quality, i.e. data quality needs CISs. Though it is possible to enact quality improvement actions by simply looking at single sources, there are some limitations that cannot be overcome by having only one copy of data to look at. For instance, accuracy improvement can be performed by having syntactic dictionaries as references; therefore, values for a field name could be quite easily syntactically checked. Nevertheless, things can become more difficult if field values cannot be compared with reference dictionaries; for instance, this is often the case of numerical data. Furthermore, even in the case in which syntactical checks can be applied, the semantic problem is not solved; for instance, if a data is syntactically correct, i.e., it respects a given format, but it is actually the wrong date, this is not easily detected by simply looking at one source. Instead, in CISs different copies of the same data are typically stored by multiple sources and can be compared in order to detect quality problems and possibly solve them.

As an example of a CIS, let us consider a set of public administrations that, in an e-Government scenario, cooperate in order to provide services to citizens and businesses. To such a scope, many data regarding both citizens and businesses are typically replicated in different administrations. Due to data errors, conflicts on data may arise that need to be solved. In Italy, an e-Government project, called “Services to Businesses” [3] has been carried on with the specific aim of reconciling conflicts among three different public administrations. The limit of this project was to adopt a centralized architecture and a considerable process re-engineering. Conversely, a more effective solution to address data quality problems in CISs requires to be distributed in order to be as less invasive as possible and to avoid bottlenecks.

1.1 The DaQuinCIS Project and Contents of the Paper

The DaQuinCIS project 1 proposes an architecture based on peer-to-peer services for quality improvement and maintenance in Cooperative Information Systems. In this architecture heterogeneous and geographically distributed organizations may exchange data and related quality data using a common semi-structured data model based on XML. The DaQuinCIS systems offers a suite of data quality oriented services that allow to exchange data and related quality, to measure the quality of data, to certify quality levels and to be notified when changes occur related to quality levels of specific data sets. The interested reader may find further details on the DaQuinCIS system in [15],[6],[23], [7].

In this paper, we propose a strategy for quality improvement in the DaQuinCIS system, called DaQuinCIS approach. The basic idea of the approach is to run a record matching algorithm periodically in order to reconcile conflicting copies on the whole CIS. Nevertheless, the extensive running of record matching algorithms is performed in fixed instants. This allows for periods in which the quality of data can deteriorate, while no quality improvement action is performed on data. Therefore, the DaQuinCIS approach proposes to complement a periodical record matching activity with an “online” quality improvement, performed at query processing time.

The paper is organized as follows. Section 2 describes the technical detail of the DaQuinCIS approach. Section 3 outlines the architecture supporting the implementation of the DaQuinCIS approach. Section 4 describes experimental results. Section 5 discusses related work, and Section 6 describes some additional issues and concluding remarks.

1“DaQuinCIS - Methodologies and Tools for Data Quality inside Cooperative Information Systems” (http://www.dis.uniroma1.it/dq/).
2. The DaQuinCIS Approach to Quality Improvement

The DaQuinCIS approach proposes to improve data quality in two distinct steps:
(i) Periodical record matching: a record matching algorithm is run on data sets stored by the organizations in the CIS. The analyzed data sets are the ones that cooperating organizations exchange with each other within cooperative processes to which they participate.
(ii) Query-time improvement: data comparisons are performed at query processing time on data received as answers to queries. When organizations request data, a component of the DaQuinCIS architecture, called Data Quality Broker, queries all the sources that may provide parts of the answer. This means that different but semantically equivalent copies of the same data may be gathered. Copies are then compared and only best quality results are returned.

Providing best quality results as answers to queries avoids the spread of low quality data throughout the system. In addition to this, best quality results may be used to obtain an improvement of the overall quality of the system. Indeed, data sources that have supplied lower quality data can be notified with the best quality results constructed during query processing.

By combining these two steps into an integrated strategy, the DaQuinCIS approach allows to improve data quality in a continuous fashion. The query-time quality improvement is first introduced in this paper, and adds a semantics to perform quality improvement to the Data Quality Broker component of the DaQuinCIS system. The Data Quality Broker is in essence a peer-to-peer data integration system. The quality improvement functionality is embedded in the queryprocessing and implies the definition of a mapping of the global schema concepts in terms of concepts at local sources in order to retrieve all copies of the same data in the system, compare and improve them. The query-time quality improvement uses the record matching algorithm in order to perform comparisons on results of queries. Section 2.3 describes the newly introduced quality improvement functionality of the Data Quality Broker. The further Data Quality Broker functionality, namely the periodical record matching, is performed by an algorithm which is is summarized in Section 2.2 and was first proposed in [2].

2.1 Basic Assumptions

As the focus of this paper is on the data quality improving strategy and on its validation, we adopt some simplifying assumptions with respect to the general setting of the DaQuinCIS system.

We assume that relational data sources exchange data with each other. We consider sources having all the same schema, though their extensions are partially overlapping. Thus, the system exhibits a certain rate of data replication. Let us notice that we do not have redundancy on data, but we assume that replicated data usually differ each other due to data errors. Data at sources have quality values associated with them. We hypothesize that each value of each attribute of source relational schemas has an associated set of quality values. We do not provide the details of how evaluating quality values; the DaQuinCIS systems includes a component called Quality Factory that is in charge of associating quality values to application data exported by cooperating organizations (see [6] for details). Let us also notice that data quality measurement is a largely investigated research field [27].

For the sake of simplicity, we choose a simplified data quality model. Specifically, let us consider a relational schema $S = \{A_1, ..., A_n\}$, where $A_i$ are the attributes of $S$. Let $\text{Quali} = \{D_i, D_o\}$ be a relation schema representing the set of quality dimensions associated to each $A_i$; we define the schema enriched with quality information as $\text{SQual} = S \cup \text{Quali}$. The DaQuinCIS system is based on more general assumptions, namely: (i) model and schema heterogeneity of data sources is considered; (ii) a more complex data model is introduced to represent data and their quality, based on the XML data model (see [23] for details).

2.2 Record Matching

Record Matching, also known as Record Linkage [8] or Object Identity problem [25], is the problem of identifying if two records are related to the same real world entity. Sets of tuples referring to semantically by the same global relation), may contain distinct copies of the same data with different quality levels, i.e., there are instance-level conflicts. In order to resolve these conflicts, it is first necessary to identify tuples which refer to the same real-world object.

In the DaQuinCIS approach, we propose to perform a record matching activity in two phases:

- a periodical record matching can be run in order to align different copies of the same entities that are present in data sources. The algorithm runs on data stored on distributed sources, according to a policy based on [9];
- record matching also supports the query processing phase by identifying same instances in query results returned by each data source (this step will be detailed in the next section).

Record matching is implemented on the basis of quality data exported by cooperating organizations. The proposed method is an improvement of the Sorted Neighborhood Method (SNM) [10], and was first proposed in [2]. The SNM consists of three distinct steps:

- Choice of the matching key. A key needs to be chosen in order to sort records that are going to be matched. The choice of the key is a fundamental step of the matching process, as the results of the matching depend on how much potentially matching records are close to each other after the sorting step.
- Sorting of records according to the chosen key.
- Moving of a fixed size window through the list of records and comparisons only of the records included in the window. Each couple of records in the window is compared in order to decide if the two records match or not.

The algorithm we propose to use within the DaQuinCIS approach is different from SNM with reference to the following aspects:

- Automation of the matching key choice phase. This phase is typically realized by a human expert, known as key designer. Instead, we suggest using quality data exported by organizations in order to select such a key.
- The decision about the matching of two records is taken in a domain independent way by considering a function that normalizes a classic edit distance function upon string lengths.

2.3 Automatic Choice of the Matching Key

We propose to exploit quality data exported by each cooperating organization in order to automatically choose the matching key. The idea is to choose a high “quality” key. Let us consider as an example the choice of a key with a low completeness value; after a sorting on the basis of such a key, the potential matching records can be not close to each other, due to null values. Similar considerations can be made also in the case of low accuracy or low consistency of the chosen key; a low accurate or low consistent key does not allow to have potential matching records close to each other. Therefore, we evaluate the quality of the matching key in terms of accuracy, consistency and completeness.

Besides quality of data, the other element influencing the choice of the key is the identification power. Let us considering as an example a record Citizen with fields Surname, Name, Address and Sex. Though the field Sex may have the higher quality value, it would not be appropriate as matching key; in fact, having only two possible values, i.e. Male and Female, all records would be simply grouped into two sets, without reaching the objective of having similar records close to each other.

On the basis of such considerations, we introduce a parameter called Identification Power of the field $j$ ($IP_j$), in order to evaluate the discriminating power of record fields.

Definition 1. $eqj$ Relation

Given two records $r_1$ and $r_2$, and given a field $j$ of the two records, we define the equivalence relation $eqj$ such that:

$$r_1 eqj r_2 \text{ iff } r_1 . j = r_2 . j$$

i.e. the value of the field $j$ of the record $r_1$ is equal to the value of the field $j$ of the record $r_2$.

Definition 2. Identification Power ($IP_j$)

The Identification Power $IP_j$ of the field $j$ is defined as:

$$IP_j = \frac{\text{Number of eqj Classes}}{\text{Total Number of Records}}$$

where $eqj$ Classes are the equivalence classes originated by the relation $eqj$ applied to the totality of records.
Besides considering the identification power of a single field, it is also possible to consider the identification power of sets of fields, for which the given definition can be easily extended. Notice that it is trivial to show that the identification power of the fields $(x,y)$ is equal to the identification power of the field $y(x)$. The choice of the matching key is performed on the basis of a function that combines quality of fields with their identification power.

The data quality parameter called Data Quality of the field $j$ (DQ$_j$) represents an overall quality value for the field $j$ and can be calculated in different ways. As an example, in [2], we calculate (DQ$_j$) as a linear combination of accuracy, consistency and completeness values for the field $j$, where the coefficients were experimentally determined. Given the overall quality value DQ$_j$ and the identification power IP$_j$, we introduce the function K$_j$, such that:

$$K_j = DQ_j \cdot IP_j$$

Let us consider all the fields $j$ of records, the steps to calculate the matching key are the following ones:

1. Computation of the Data Quality of the field $j$.
3. Computation of the function $K_j$.
4. Selection of the matching key as $\max K_j$.

The selection of a set of fields to construct the key is also possible and the computation of the Data Quality and the Identification Power can be easily extended to such cases.

### Matching Decision

The method we propose for matching decision is based on a specific edit distance function; string or edit distance functions consider the amount of difference between strings of symbols. We have chosen the Levenshtein distance [14], which is a well known early edit distance where the difference between two text strings is simply the number of insertions, deletions, or substitutions of letters to transform one string into another. The function we use for deciding if two strings $S_1$ and $S_2$ are the same is also dependent from the lengths of the two strings as follows:

$$f(S_1, S_2) = \max(\text{length}(S_1), \text{length}(S_2)) \cdot \frac{\text{LD}(S_1, S_2)}{\max(\text{length}(S_1), \text{length}(S_2))}$$

According to such a function, we normalize the value of the Levenshtein distance by the maximum between the lengths of the two strings, i.e. the function $f$ is 0 if the strings are completely different, 1 if the strings are completely equal.

The procedure we propose to decide if two records match each other is the following:

- The function $f$ is applied to the values of a same field in the two records. If the result is greater than a fixed threshold $T_1$, the two values are considered equal; we call $T_1$ the field similarity threshold.
- If the number of equal pairs of values in the two records is greater than a threshold $T_2$, then the two records are considered as match; we call $T_2$ record similarity threshold.

The thresholds $T_1$ and $T_2$ have to be fixed experimentally. Specifically, if choosing an exact matching policy, $T_1 = 1$ and $T_2 = \text{Total number of fields composing records}$. Otherwise, if choosing an approximate matching policy, the two thresholds can be adjusted according to the required strictness of the matching policy.

We considered recall and precision metrics for the proposed record matching algorithm on experimental data sets. The results are described in [2] where we show the effectiveness of the method, also compare to the basic Sorted Neighborhood Method. The final result of the record matching algorithm is a partition of records into a set of clusters, each one consisting of records referring to the same real world object, as detailed in the following section.

### 2.3 Query-time Quality Improvement

Data exchanges between data sources are performed through the Data Quality Broker of the DaQuinCIS architecture. The Data Quality Broker acts essentially as a mediation layer that performs data integration. Queries on the global schema are processed according to a global-as-view (GAV) approach by unfolding, i.e., replacing each atom of the original query with the corresponding view on local data sources [24, 13]. The specific way in which the mapping is defined stems from the idea of performing a quality improvement function during the query processing step. While the model and query processing techniques are detailed in other works ([23, 17]), in this paper we focus on how the improvement functionality of the broker is implemented. When defining the mapping between concepts of the global schema and concepts of the local schemas, each concept from the global schema is defined in terms of extensionally overlapping concepts at sources. Therefore, when retrieving data, they can be compared and a best quality copy can be constructed. Specifically, data sources have distinct copies of the same data with different quality levels. We resolve these conflicts at query execution time by relying on quality values associated to data: when a set of different copies of the same data are returned, we look at the associated quality values, and we construct the copy to return as a result on the basis of such values. The best quality copy is also diffused to other organizations in the CIS as a quality improvement feedback. The algorithm to answer a query and construct the best quality result is exemplified in Figure 1.

The first step is the unfolding of a query $Q$ over a global schema. The query $Q$ is unfolded according to the static mapping that defines each concept of the global schema in terms of the local sources; such a mapping is defined in order to retrieve all copies of same data that are available in the CIS. Therefore, the query $Q$ is decomposed into $Q_1, \ldots, Q_t$ to be posed over local sources. Such queries are then executed to return a set of results $R_1, \ldots, R_t$. On such a set an extensional correspondence is checked. Specifically, the record matching algorithm described in the previous section is run on the set $R_1 \cup R_2 \cup \ldots \cup R_t$. The result of the running of the record matching algorithm is the construction of a set of clusters composed by records referring to same real world objects, namely $C_1, \ldots, C_q$. The result to be returned is built by relying on a best quality default semantics. For each cluster, a best quality representative is either selected or constructed. Each record in the cluster is composed by couples in which a quality value $q$ is associated to each field value $f$. The best quality record for each cluster is selected as the record having the best quality values on all fields, if this record exists. Otherwise, such a best quality record is constructed by composing the fields that have the highest quality from records within the same cluster. Once representatives for each cluster have been selected, the result $R$ is constructed as the union of all cluster representatives.

Let us note that each quality value $q$ is a vector of quality values corresponding to the different quality dimensions. For instance, $q$ can include values for accuracy, completeness, consistency and currency. These dimensions have potentially different scales, therefore a scaling problem occurs. Once scaled such vectors need to be ranked, therefore also a ranking method must be applied. Both scaling and ranking problems have well-known solutions (e.g., multi-attribute decision making methods, like AHP [21], see also later in this section). The process that leads to the building of the results is shown in Figure 2.

Beside providing $R$ as a result to the query $Q$, $R$ is also proposed as the best quality copy available in the CIS to organizations, having provided lower quality results. Organizations involved in the query have the choice of updating or not their data. This gives a non-invasive feedback that allows to enhance the overall quality of data in the CIS, while preserving the autonomy of cooperating organizations.

Notice that the Data Quality Broker is a notable example of a system performing quality improvement in a virtual data integration settings. Indeed, data quality improvement activities have been mainly considered in materialized data integration contexts, e.g. data warehouses. Nevertheless, increasing attention is being paid to data reconciliation and improvement also within virtual data integration contexts (see Section 5).

### 3. The Reference Architecture for the DaQuinCIS Approach

In this Section, we describe the architectural features of the Data Quality Broker, which is the component of the DaQuinCIS system that allows the actual implementation of the DaQuinCIS approach.
The Data Quality Broker is implemented as a peer-to-peer distributed service: each organization hosts a copy of the Data Quality Broker that interacts with other copies (see Figure 3, left side). Each copy of the Data Quality Broker is internally composed by four interacting modules (see Figure 3, right side).

The Query Processor performs query processing, as detailed in Section 2.3. It unfolds queries posed over the global schema and passes local queries to the Transport Engine module. On receiving query results, a query refolding phase is performed, in order to make the execution of the global query possible. A record matching activity is then performed in order to identify all copies of same data returned as results. The record matching algorithm is also periodically run by the Query Processor on the whole data in the system. In this implementation, we have forced one organization to be responsible of such peridical task, leaving to future work possible extensions of this solution. Then, matched results are ranked on the basis of associated quality by the Comparator module. Finally, the Query Processor selects the best quality copy(ies) to be returned as a result and also to be sent as improvement feedbacks to organizations having provided lower quality data. The module Comparator is used by the Query Processor in order to compare different quality values and construct the best quality representative record from those gathered as query results. The Transport Engine is a communication facility that transfers queries and their results between the Query Processor module of a source and other data sources. The module Propose Manager is in charge of managing data quality improvement feedbacks that are issued by other organizations at query time.

The Query Processor is responsible for query execution. The copy of the Query Processor local to the user query receives the query and splits it into queries local to the sources, on the basis of the defined mapping. Then, the local Query Processor also interacts with the local Transport Engine in order to send local queries to other copies of the Query Processor and receive the answers. The Transport Engine provides general connectivity among all Data Quality Broker instances in the CIS. Copies of the Transport Engine interact with each other in two different scenarios:

– Query execution: the requesting Transport Engine sends a query to the local Transport Engine of the target data source by executing the query() operation (see Figure 3) and asynchronously collects the answers.
– Quality feedback: when a requesting Transport Engine has selected the best quality result of a query, it contacts the local Transport Engine to enact quality feedback propagation. The feedback() operation (see Figure 3) is executed as a callback on each organization, with the best quality selected data as a parameter.
3.2 The System Architecture
The Data Quality Broker system architecture is based on web services technologies. To implement web services, we have chosen the J2EE 1.4 Java Platform, specifically the Java API for XML-based Remote Procedure Call (JAX-RPC) [12]. In JAX-RPC, request/response of remote methods is performed through the exchange of SOAP messages over an HTTP connection. Each copy of the Data Quality Broker is implemented by two web services, namely the Query Processing Web Service and the Data Manager Web Service as shown in Figure 4. In the figure, the query) operation is executed by a client with a user query as input, thus making the Quality Processing Web Service call the query() operations, executed by the Data Management Web Services of the remote copies of the Data Quality Broker.

4 Experiments
In order to verify the effectiveness of our data quality improvement approach, we performed experiments with large real data sets. In this Section, we first show the adopted experimental methodology in Section 4.1; then, in Sections 4.2 and 4.3, we show respectively the results of quality improvement experiments and performance experiments.

4.1 Experimental Methodology
We performed two sets of experiments. The first set shows the effectiveness of the DaQuinCIS approach to improve data quality. The second set shows the performance features of the DaQuinCIS system, when running the proposed improvement strategy.

Figure 4. The Data Quality Broker system architecture

In both cases, a comparison is described with a "standard" system, which does not include any data quality management feature. Conversely, the DaQuinCIS system adopts the described quality improvement strategy.

We used two real data sets. Each data set is owned by an Italian public administration agency, namely:
- The first data set is owned by the Italian Social Security Agency, referred to as INPS (in Italian, Istituto Nazionale Previdenza Sociale). The size of the database is approximately 1.5 millions of records.
- The second data set is owned by the Chambers of Commerce, referred to as CoC. The size of the database is approximately 8 millions of records.

Some data are agency-specific information about businesses (e.g., employees social insurance taxes, tax reports, balance sheets), whereas others are common to both agencies. Common items include one or more identifiers, headquarter and branches addresses, legal form, main economic activity, number of employees and contractors, information about the owners or partners. We have calculated the degree of overlapping of the two databases that is equal to about 970000 records. The experimental setting consists of three data sources. Two of these data sources correspond to the above described data sets. The third one is only used to take into proper account the effect of queries between sources. We consider that it initially contains no data and that all data requests are issued by this particular data source. Each time this source poses a query, it stores the results. As detailed in the following, we measure quality referring to the data stored in the system as a whole.

for the frequency of queries and updates on the databases as well as the average query result size are derived from real use cases. The frequency of changes in tuples stored in the two databases are estimated to be around 5000 tuples per week for each database. In a real setting, updates are almost uniformly distributed over a week. Nevertheless, in order to simplify our experimental setting and to better distinguish between changes in quality due to updates and changes due to internal data exchanges, we have chosen to limit updates to the beginning of each week. Average query frequency and query result size are, respectively, 3000 queries per week and 5000 tuples per query. The queries may be assumed to be uniformly distributed over the entire week.

We have associated data quality values to the the INPS and CoC databases. We have chosen to use two data quality dimensions, namely completeness and currency. Data quality values are associated with each value of each attribute of the databases. Completeness refers to the presence of a value for an attribute, therefore completeness values are 0 if no value is present, 1 otherwise. As far as currency values, tuples of the two experimental data sets had associated boolean currency values, according to a quality check that had already been performed on the databases.

In order to measure the quality of the system we adopted a simplified quality metrics, considering that a tuple has high quality if it is complete and current on all its attributes. Conversely, a tuple has low quality if it is not complete and/or current on some attributes. More formally, given the relational schema

\[ SQ_{Qual} = \{ A_1, \ldots, A_n \} \]

where the first addend considers changes in quality related to updates, whereas the second one is related to the overall quality of query results. Notice that both these contributions may be positive or negative. In particular, the sign of the first addend depends on the quality of tuples updated in the system, while the sign of the second one depends on how queries cause data of bad or good quality to spread into the system. To clarify the relations between the standard and the DaQuinCIS system, we can further split the quality check into two terms referring respectively to query results coming from semantically-overlapping and non-semantically-overlapping parts of the databases:

\[ \Delta Q_{System} = \Delta Q_{Updates} + \Delta Q_{Queries} \]

where the first addend considers changes in quality related to updates, while the second one is related to the overall quality of query results. Notice that both these contributions may be positive or negative. In particular, the sign of the first addend depends on the quality of tuples updated in the system, while the sign of the second one depends on how queries cause data of bad or good quality to spread into the system. To clarify the relations between the standard and the DaQuinCIS system, we can further split the quality check into two terms referring respectively to query results coming from semantically-overlapping and non-semantically-overlapping parts of the databases:

\[ \Delta Q_{DaQuinCIS} = \Delta Q_{Updates} + \Delta Q_{Queries} + \Delta Q_{RecordMatching} + \Delta Q_{Query-time} \]
\( \Delta Q_{\text{Updates}} \), as in the standard system, represents the variation coming from updates of tuples in the databases. As in the standard system, this addend may be either positive or negative. The variation coming from queries is reduced here to the part related to non-semantically-overlapping portions of the databases. Also this contribution may represent either an improvement or a decrease of the overall quality. The other two contributions, on the other hand, always represent data quality improvements. We emphasize this fact by a small ‘+’ near the \( \Delta \) symbol.

\( \Delta - Q_{\text{Record Matching}} \) is the percentage of tuples which are improved by the periodical execution of the matching algorithm. The term \( \Delta - Q_{\text{Query-time}} \) is instead related to the query-time improvement actions.

It accounts for two different terms, as query-time improvement depends on two distinct actions, namely the (query-time) record matching that is performed on the result of queries, and the improvement feedbacks:

\[
\Delta - Q_{\text{Query-time}} = \Delta - Q_{\text{overlapping}} + \Delta - Q_{\text{Feedbacks}}
\]

\( \Delta - Q_{\text{overlapping}} \) is the percentage of tuples which are improved due to the best quality answering semantics of the DaQuinCIS system. Let us notice that this contribution corresponds to the \( \Delta Q_{\text{overlapping}} \) of the standard system, but it is always a positive contribution, due to the specific semantics of the DaQuinCIS system.

\( \Delta - Q_{\text{Feedbacks}} \) is an additional improvement related to the propagation of best quality results to data sources. We are assuming here that proposed best quality feedbacks are always accepted by sources. This hypothesis does not hold in the general DaQuinCIS setting.

We can now express the overall gain in quality improvement that is obtained in the DaQuinCIS system as:

\[
\Delta - Q_{\text{DaQuinCIS}} - \Delta - Q_{\text{Standard}} = \\
\Delta - Q_{\text{Record Matching}} + \Delta - Q_{\text{Query-time}} - \Delta - Q_{\text{Queries}} = \\
\Delta - Q_{\text{Record Matching}} + \Delta - Q_{\text{Feedbacks}} + \left( \Delta - Q_{\text{overlapping}} - \Delta - Q_{\text{overlapping}} \right)
\]

Notice that the term in parentheses is always positive, since:

\[
\Delta - Q_{\text{overlapping}} \leq \Delta - Q_{\text{overlapping}}
\]

4.2 Quality Improvement Experiments

To investigate the effectiveness of the DaQuinCIS approach with respect to quality improvement, we have considered three different scenarios:

- **Case 1** A fixed number of tuples is updated with high quality tuples.
- **Case 2** A fixed number of tuples is updated with low quality tuples.
- **Case 3** Cases 1 and 2 are mixed, using an update set composed by both high quality and low quality tuples equally distributed.

In each of the three cases, the evolution of the overall quality level in the DaQuinCIS and standard system is analyzed within a period of five weeks, during which the data sources exchange data and receive updates with the frequencies above described.

In the following of this section we will discuss the results we obtained for each of three cases, comparing the different trends exhibited by the two systems.

In **Case 1**, a fixed number of tuples of the system are updated each week with high-quality tuples. Figures 5 and 6 show the behavior of the two systems under this experimental condition. In the figures, the overall Quality of the system is drawn with respect to time. Before analyzing the details of the two figures, it is worthwhile to note two things. First, the two curves have steps at the beginning of each week. This is due to the simplifying assumption of considering updates at the beginning of each 7 days period. Second, the curve that refers to the DaQuinCIS system starts from a percentage of quality equal to about 85.3%, while the standard system curve starts from a percentage of about 79.7%. The difference is due to an initial record matching performed in the DaQuinCIS system. Therefore, it is immediately shown the improvement effect of the matching, considering that the two systems start from initial quality levels that are identical. Looking at Figure 5, in the standard system, the change of quality in each period between updates is only due to queries. The picture shows a very low, roughly linear growing trend. This is due to the slow spread of the high quality tuples, updated at the beginning of each week, throughout the system. The overall quality improvement during the five weeks period is

\[
\Delta Q_{\text{Standard}} = \text{Final Quality} - \text{Initial Quality} = (80.59\% - 79.68\%) = 0.91\%
\]

With reference to Figure 6, in the DaQuinCIS system the improvement of quality in each period is enhanced by the query time quality actions. Firstly, only best quality results are returned as answer to queries. This feature alone would be sufficient to have a higher growing trend with respect to the standard system. Secondly, high quality tuples are diffused throughout the system by improvement feedbacks. The overall trend resulting from these two concurrent actions is much more rapid than in the standard system. In order to calculate the total improvement in the DaQuinCIS case, we must separately take into account two distinct contributions. The first one due to the record matching performed at the beginning of the observed period. It improves the quality of the system from the initial quality value of 79.68% to a final value of 85.30%. Thus \( \Delta Q_{\text{Record Matching}} = \text{Quality after Record matching} - \text{Initial Quality} = (85.30\% - 79.68\%) = 5.62\% \). The second contribution is due to updates and queries, as in the standard case, with the additional improvement actions performed in DaQuinCIS at query-time. Its value is \( \Delta Q_{\text{Other DaQuinCIS}} = \text{Final Quality} - \text{Quality after Record Matching} = (87.09\% - 85.30\%) = 1.79\% \). Notice that the query-time contribution alone is already two times the quality improvement obtained in the standard system. The overall variation of quality is then

\[
\Delta Q_{\text{DaQuinCIS}} = \Delta Q_{\text{Record Matching}} + \Delta Q_{\text{Other DaQuinCIS}} = 7.41\%.
\]

The actual gain in improvement due to the DaQuinCIS system is the difference \( \Delta Q_{\text{DaQuinCIS}} - \Delta Q_{\text{Standard}} = 6.5\% \). As the initial overall size of the two database is 9.5 millions tuples, this means that the DaQuinCIS system improves the quality of approximately 620000 tuples more than the standard system. Figure 7 graphically shows the
difference between the two systems.

**Case 2** is represented in Figures 8 and 9: a fixed number of tuples of the system are updated with low-quality tuples at the beginning of each week. In the standard system, represented in Figure 8, queries cause the spread of low quality values that lower the overall quality of the system.

In the DaQuinCIS system, represented in Figure 9, the effect of an initial record matching makes the DaQuinCIS curve start with a higher quality level, as in the previous case. Furthermore, the spread of low quality tuples related to queries is prevented, and the improvement action based on feedbacks counterbalances the loss of quality due to updates. Indeed, whenever low quality tuples are involved in a data request, it may happen that they are substituted by high quality tuples due to improvement feedbacks.

As in the previous case, we can calculate \( \Delta Q_{\text{Standard}} \) and \( \Delta Q_{\text{DaQuinCIS}} \). It results that \( \Delta Q_{\text{Standard}} = \text{Final Quality} - \text{Initial Quality} = (78.82\% - 79.68\%) = -0.86\% \), i.e.

![Figure 5](image)

**Figure 5.** Data quality improvement in the standard system: high quality case

![Figure 6](image)

**Figure 6.** Data quality improvement with DaQuinCIS: high quality case

In **Case 3**, we consider how the DaQuinCIS and the standard system evolve under the more general hypotheses of updates including both high and low quality tuples. This case is illustrated in Figures 10 and 11. While the number of updated tuples per week does not change, a half of the updated tuples are of high quality and the other half of low quality. In this case, changes in quality at each update are not perceivable. Thus, the two curves do not exhibit the steps that characterized them in the two previous cases. This also causes a lower range in the two graphs, so we have slightly changed the scale used in the representation with respect to the previous cases. Consistently with previous observations, the quality remains approximately constant in the standard system (see Figure 10). Conversely, in the DaQuinCIS system, there is an increase in quality of more than half percentage point, in addition to the initial increment which is, as usual, related to the initial record matching. More precisely, \( \Delta Q_{\text{DaQuinCIS}} = \text{Final Quality} - \text{Quality after Record Matching} = (85.30\% - 85.30\%) = 0.59\% \). The improvement due to the initial record matching is of course still equal to 5.62%, so \( \Delta Q_{\text{DaQuinCIS}} = (5.62\% + 0.59\%) = 6.21\% \). As \( \Delta Q_{\text{Standard}} \) is approximately equal to zero, the difference in improvement between the two systems \( \Delta Q_{\text{DaQuinCIS}} - \Delta Q_{\text{Standard}} = +6.21\% \).

![Figure 7](image)

**Figure 7.** Data quality improvement in DaQuinCIS case vs. standard case

![Figure 8](image)

**Figure 8.** Data quality improvement in the standard system: low quality case

![Figure 9](image)

**Figure 9.** Data quality improvement with DaQuinCIS: low quality case

![Figure 10](image)

**Figure 10.** Data quality improvement in the standard system: mixed quality case
As a further experiment, we show the detailed behavior of the DaQuinCIS query time improvement when considering a single period and varying the number of performed queries and the size of the result. This curve better shows the typical trend of the DaQuinCIS improvement due to query time improvement actions. In Figure 12, the number of improved tuples within a period is shown. By increasing either the size of the result or the query frequency, the curve grows more rapidly.

Figure 11. Data quality improvement with DaQuinCIS: mixed quality case

Figure 12. The DaQuinCIS improvement at query time within a single period while varying query size and query number

4.3 Performance Experiments

For performance experiments, a distributed environment has been simulated. Each data source has been deployed on a different computers, connected by a LAN at 100 Mbps.

We have considered the DaQuinCIS system and the standard system behavior with fictitious sources, in order to vary some parameters influencing performance experiments. The first performance experiment shows the time overhead of the DaQuinCIS system with respect to the standard system. In such experiment, we draw a normalized transaction time defined by the fraction:

\[
\frac{\text{DaQuinCIS Elaboration Time}}{\text{Standard Elaboration Time}}
\]

The elaboration time is the time required by the system for processing a query. The normalized transaction time is drawn when varying the degree of overlapping of data sources. The overlapping degree is an important feature of the DaQuinCIS system. Indeed, the DaQuinCIS system accomplishes its functionalities in contexts where data sources overlap and such an overlapping can be exploited to improve the quality of data. The Figure 13 shows how the normalized transaction time varies in dependence on the percentage of data sources overlapping with two fixed query result sizes, namely q1=1000 tuples, q2=5000 tuples. The number of overlapping sources is fixed to 3. This means that once a query is posed over the system, three sources have data that can be provided as an answer to the queries, though the system can have a larger number of sources. The Figure 13 shows the actual time overhead of the DaQuinCIS systems with respect to a standard system. The DaQuinCIS system has an acceptable time overhead. The worst depicted case is for the query result size q2=5000 and a percentage of overlapping equal to 40%; in such a case, there is a 50% time overhead with respect to the standard system.

Notice that a degree of overlapping around 15% is quite typical in e-government settings where public agencies that store data have only recently started to cooperate and still maintain data acquired when they operated in an autonomous way. Therefore, in this setting, the DaQuinCIS system overhead with respect to the standard system is typically lower than 20%. Nevertheless, as shown by the improvement experiments described in Section 4.2, an overlapping rate of about 12% still allows to have good results in terms of quality improvement effectiveness.

The second performance experiment shows the normalized transaction time with query size varying (see Figure 14). For a fixed degree of overlapping equals to 15%, we draw the normalized transaction time for three different numbers of overlapping organizations, namely n1=3, n2=4 and n3=5. This experiment shows the behavior of the DaQuinCIS system when increasing the number of organizations and the size of queries. Specifically, the normalized transaction time increases slowly with an almost linear trend. The positive result shown in figure 14 is that when the number of overlapping data sources increases, the trend does not substantially change.

Figure 13. Normalized transaction time wrt percentage of overlapping data sources

Figure 14. Normalized transaction time wrt query sizes

5 Related Work

Data quality improvement is an important task when performing data integration [20]. Most of the work concerns quality improvement of data in materialized settings; for instance, when building a data warehouse system, an expensive and time-consuming cleaning activity needs to be performed [11].

Instead, very few works address the data quality improvement problem in a virtual data integration setting. In [19], an algorithm for querying for best quality data in a LAV integration system is proposed. The mapping between the local schemas and the global schema is expressed by means of assertions called Query Correspondence Assertions (QCA’s). Quality values are statically associated to QCA’s and to data sources. Instead, some quality values are associated to user queries at query time. In the DaQuinCIS framework, we share with [19] the idea of querying for best quality data. However, the main difference of our work with respect to [19] is the semantics of our system. Our aim is not only querying, but also improving quality
of data. To such a scope, the query processing step has a specific semantics that allows to perform quality improvement on query results.

The MIT Context Interchange project (COIN) [5] is based on the idea of modeling a "context" for integrating heterogeneous sources. Such a context consists of metadata that allows for solving problems, such as instance-level conflicts that may occur in the data integration phase. An integration system based on the LAV approach implements query processing with contexts. The DaQuinCIS approach differs mainly for considering a much more general and explicit way of representing quality of data. Instead, the COIN approach focuses only on one aspect of data quality, namely data interpretability.

In [16], the basic idea is querying web data sources by selecting them on the basis of quality values provided on data. Specifically, the authors suggest to publish metadata characterizing the quality of data at the sources. Such metadata are used for ranking sources, and a language to select sources is also proposed. In the DaQuinCIS system, we associate quality to data (at different granularity levels) rather than to a source as a whole. This makes things more difficult, but allows to pose more specific queries. For instance, the DaQuinCIS system easily treats the quite common cases in which a source has some data which have a low quality and some other ones that have instead a higher quality, by making the source be an actual data provider only for better quality data. No improvement feature is present in [16].

Furthermore, some systems are focused on data reconciliation and merging within a data integration setting [4]. The data reconciliation phase deals with problems at instance level, whereas data conflicts have to be resolved. Examples of data integration systems dealing with instance level conflicts are the interactive system presented in [22] and the AURORA system proposed in [28]. The system described in [22] describes how to solve both semantic and instance-level conflicts. The proposed solution is based on a multidatabase query language, called FraQL, which is an extension of SQL with context resolution mechanisms. AURORA supports conflict tolerant queries, i.e. it provides a dynamic mechanism to resolve conflicts by means of defined conflict resolution functions. In the DaQuinCIS system, we introduce a set of quality values associated to data at each source; then we can rely on such values when performing a comparison of different copies of data, thus adopting a quite different approach for solving multi-source conflicts.

In 1999 the Italian Public Administration started a project, called "Services to Businesses", which involved extensive data reconciliation and cleaning [3]. The approach followed in this project consisted of three different steps: (i) linking once the databases of three major Italian public administrations, by performing a record matching process; (ii) correcting matching pairs and (iii) maintaining such a status of aligned records in the three databases by centralizing record updates and insertions only on one of the three databases. This required a substantial re-engineering of administrative processes, with high costs and many internal changes for each single administration. Differently from the approach adopted in the "Services to Businesses" project, the DaQuinCIS approach does not create any bottleneck on a single cooperating organization, since the improvement is managed in a complete distributed way. Moreover, no kind of re-engineering actions need to be engaged, as the quality improvement actions are enacted according to a non-invasive policy.

6 Concluding Remarks

The paper provides two major contributions. The first contribution is the definition and implementation of a quality improvement functionality. This is performed at query processing time and has been implemented in a distributed fashion. According to our knowledge, this is the only proposal that exploits data replication in a cooperative system in order to improve the quality of data. The quality improvement functionality uses a record matching algorithm which is fully automatic. Although such an algorithm is not a contribution of this paper, the usage of the algorithm to perform query-time improvement has been first introduced here. The complete process that enables query time quality-improvement starting from the definition of a "quality supportive" mapping and ending with a best quality provided result is also a contribution of this paper.

The second contribution is an overall approach to data quality improvement in cooperative information systems, in which a periodical record matching activity is complemented by a quality improvement performed at query processing time. The approach and its effectiveness for quality improvement are experimentally validated in a real setting. Moreover, some performance measures also show efficiency features. The usage of the DaQuinCIS system implies some trade-offs that cooperating organizations have to agree upon: the availability of the best quality copy of some data can be guaranteed at the cost of some deterioration in performances, although controlled and limited. This trade-off inhibits the usage of the DaQuinCIS approach in scenarios with strict performance requirements, but its usefulness in many other scenarios can be proved: as an example, for e-government applications the effectiveness of service provision is much more important than system performances [1].

In the future, we will investigate how trust issues can be integrated within the query processing phase of the Data Quality Broker. Specifically, in the DaQuinCIS system there is already a component called Rating Server that provides a dynamic assignment of trust values to each cooperating organization (see [7] for details). The idea is to use such trust values in order to weight data quality vectors properly when selecting best quality results.

As a further issue, we plan to consider the behavior of the DaQuinCIS system with real data belonging to different business domains and compare the different behaviors. Indeed, some parameters such as data update frequency, record matching periodicity, query frequency could be related to each other, in order to formulate a generalized approach for quality improvement to be customized for every kind of domain.

References