

Managing Data Quality and Integrity in Federated Databases

Michael Gertz

*Department of Computer Science, University of California, Davis
One Shields Avenue, Davis, CA 95616-8562, USA*

Phone/Fax: +1-530-762-6468/-4767

e-mail: gertz@cs.ucdavis.edu

Abstract

Today many organizations are facing the need to integrate multiple, previously independent but semantically related information sources. Although often the quality and integrity of the data at a single information source itself is a problem, issues and problems regarding the quality and integrity of integrated data have been mainly neglected by approaches to data integration. As recent studies show, this raises severe problems for global applications that depend on the reliability of the integrated data.

In this paper we focus on problems and possible solutions for modeling and managing data quality and integrity of integrated data. For this, we propose a taxonomy of data quality aspects that includes important attributes such as timeliness and completeness of local information sources. We use the federated database approach to describe how data quality aspects can be modeled as metadata during database integration. These metadata are employed to specify data quality related query goals for global applications and to dynamically integrate data from component databases storing data of different quality. The presented concepts are based on treating data quality and integrity as a first-class concept in both metadata model and global query language.

Keywords

Database Integration, Federated Databases, Data Quality, Data Integrity, Metadata, Dynamic Data Integration

1 INTRODUCTION

In order to stay competitive and to rely on accurate and complete information, many enterprises and federal agencies are facing the need to integrate several, previously independent but semantically related information sources into globally accessible systems. Integration approaches are typically based on multidatabase or federated database systems (see, e.g., (Sheth and Larson 1990, Bright *et al.* 1992, Kim 1995)). Applications built on top of such

systems include data warehouses, decision support systems, hospital information systems, environmental information systems etc.

Recent studies and reports show that these applications, in particular data warehouses, often experience several problems with regard to the reliability of the integrated data (Kimball 1996, Bischoff and Alexander 1997, Jarke and Vassiliou 1997). The main reason for this is that often already the component databases participating in the federation contain incorrect and poor quality data. The quality and integrity of the integrated data then becomes even worse unless suitable methods and techniques for modeling and managing these issues are employed.

During the past decade a significant amount of literature on database integration has been published. Most of the proposed approach focus on the problem of resolving structural and semantic conflicts among (heterogeneous) component database. For an overview of the problems and proposed solutions see, e.g., (Kim and Seo 1991, Spaccapietra *et al.* 1992, Sheth and Kashyap 1993, Kim *et al.* 1995). Only a minor part of the literature discusses how to handle integrity constraints in data integration, mainly in connection with resolving structural and semantic conflicts, e.g., (Reddy *et al.* 1995, Vermeer and Apers 1996, Conrad *et al.* 1997).

In this paper we claim that, from a practical point of view, the traditional notion of data integrity, i.e., the semantic correctness of stored data, plays only a minor role with regard to the reliability of integrated data. Typical problems that occur at the integration level and which cannot be prevented by traditional integrity maintaining methods are rather data quality problems than data integrity problems. For example, the aspect of outdated or expired data at the integration level often is referred to as a data integrity problem. But there neither exists a formalism nor a technique to prevent the integration of outdated data. A similar assumption is made for the accuracy or completeness of data at the integration level. Again, there does not exist a data integrity concept that covers these aspects during database integration.

There are two key issues or rather assumptions in database integration approaches that contribute to the fact of poor quality and integrity data at the integration level. First, existing approaches assume that the data stored at component databases (CDBs) somehow have the same high quality and correctness. Considering individual CDBs this is true, because the data at each site are sufficient for local applications. That is, for example, while outdated data may be sufficient for a local application at one CDB, another CDB must always contain up-to-date data. Similar scenarios can be given concerning the completeness and accuracy of data, aspects which are also not considered by existing data integration approaches. Thus there is a strong need for modeling these aspects as well.

But such a modeling approach leads to the next problem. Existing data integration techniques assume that schematic and semantic conflicts can always be resolved statically by giving unique schemas for global relations and asso-

ciated data integration rules. While this often is true for schematic aspects, the data stored at different component databases typically change with time. For example, a component database may only have up-to-date data at certain days in a week. We thus cannot always integrate data (of different quality and integrity) in a static manner by using a fixed set of data integration rules.

The above aspects raise the question how we can model, manage, and represent different data quality and integrity aspects during database integration. In this paper, we describe our observations and preliminary results on studying different aspects of data quality and integrity in federated databases. The main idea is to treat data quality as a first-class concept in data integration approaches and in querying integrated data. Based on a taxonomy of (time-varying) data quality aspects, these aspects can be modeled suitably as metadata at the integration level. For global applications, the designer can specify data quality goals for global queries used in applications, and the global query processor suitably decomposes global queries into subqueries such that the retrieved data all have the same quality. This ensures that data of different quality are not combined or joined. Information about the quality of retrieved data is represented at the integration level as well, thus providing global users a suitable means to cope with incorrect and poor quality data.

The paper is organized as follows: In Section 2 we outline the main concepts and techniques of the federated database scenario that are important for the presented approach. In Section 3 we motivate the distinction between data quality and data integrity and, based on a taxonomy of data quality, we show how data quality aspects can be modeled as metadata during database integration. The usage of the metadata, which also include information about local integrity constraints, is discussed in Section 4. Finally, concluding remarks and future work is presented in Section 5.

2 THE FEDERATED DATABASE APPROACH

For our approach to manage data quality and integrity in a multidatabase environment, we adopt the schema and system architecture for federated database systems described in, e.g., (Sheth and Larson 1990). That is, we assume a global (or integrated) database schema that provides users and applications with an integrated view of all local databases schemas (or export schemas). For our discussions, we use the relational data model as the global data model because of its simple structure. The presented approach can easily be adopted to an Object-Oriented model.

The basic components in of a federated database system is the federation layer, also called integration level (see also Figure 1). Depending on the type of global applications, this layer can be itself a database system providing a global data model for schema integration, a metadata repository capturing the information related to the integration process, and a global query language and processor. The query processor utilizes the metadata to map queries

against the global schema to (sub)queries against the component databases (CDBs), and is discussed in more detail in Section 4. If global applications are read-only applications such as, e.g., data warehouses, then the above components are sufficient to build a federation layer. Data modifications through global applications additionally require a global transaction manager, e.g., (Breitbart *et al.* 1995), which will not be considered in this paper.

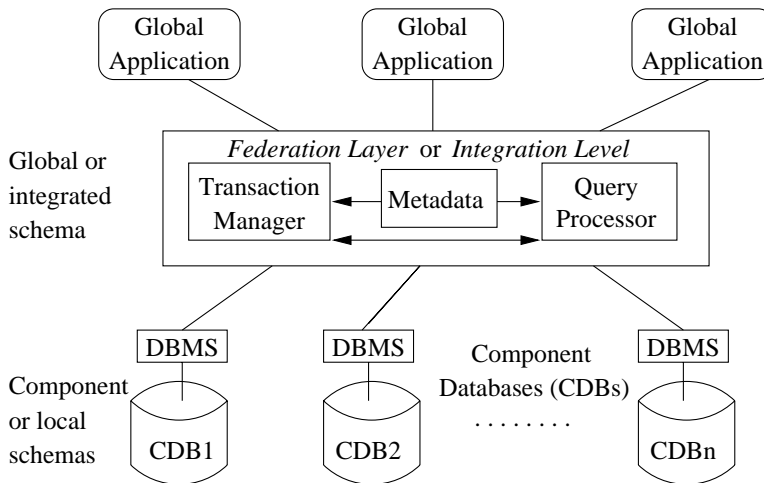


Figure 1 Schema and System Architecture of a Federated Database System

A major task in integrating data from different autonomous and possibly heterogeneous database systems is to detect correspondences among the elements of component schemas and to resolve possible conflicts (Batini *et al.* 1986, Sheth and Larson 1990, Kim *et al.* 1995). Conflicts arise due to various heterogeneity of the databases participating in a federation, ranging from technological discrepancies (software, hardware) to schematic and semantic heterogeneity. While the former type of conflicts often can be resolved using suitable protocols and wrappers, the latter type of conflicts is much more difficult to cope with and has been a major concern among the database research community (Bright *et al.* 1992, Litwin *et al.* 1990, Sheth and Kashyap 1993, Kim *et al.* 1995).

Schematic and semantic heterogeneity stems from the ability of component databases to choose their own design, including data model, query language, naming of data (relations and attributes), and in particular semantic interpretations of the data and attribute values. During the past decade a significant amount of literature discussing methods on resolving schematic and semantic conflicts has been published. These methods focus on naming conflicts (homonyms and synonyms) and in particular on semantic data conflicts (modeling the same information in different ways) (Kim and Seo 1991, Rusinkiewicz and Missier 1995, Spaccapietra *et al.* 1992).

Essential for our approach to manage data quality and integrity in federated databases is the fact that almost all approaches to resolving schematic and semantic conflicts are based on static conflict resolution and in particular data integration rules. Once semantic conflicts among data stored in component databases have been detected, they are resolved by specifying suitable conflict resolution rules or views in the metadata repository. Such rules include relation and attribute renaming, joining local relations, and data conversions between different scales and measurements. After restructuring of local relations at the federation layer, component schemas are in a form that can easily be integrated by, e.g., building the union of the restructured local relations that contain similar or related data. These views then build the elements of the global database schema and only these views are visible to global users and applications. For decomposing global queries against these views, the global query processor employs data integration and conflict resolution rules which are recorded in the metadata repository.

3 DATA QUALITY AND INTEGRITY IN DATABASE INTEGRATION

In this section we investigate the distinction between the notions of data quality and data integrity, and we examine how the underlying concepts influence traditional database integration approaches. As we will show in Section 3.1, many aspects often referred to as data integrity are basically data quality aspects. In Section 3.2 we show how to identify and describe different data quality aspects during database integration. It turns out that neglecting these aspects can have a major impact on the reliability of the integrated data. In Section 3.3 we finally describe how the proposed data quality and integrity aspects can be modeled as integration metadata, thus allowing designers of global applications and the query processor to treat these aspects as first-class concepts at the integration level.

3.1 From Data Integrity to Data Quality

As information is becoming a key organizational resource, many companies and federal agencies are concerned about the quality of the data they manage and use for their applications (Redman 1996). In the past years a significant amount of literature focusing on data quality has been published, ranging from defining frameworks of data quality aspects (Wang and Strong 1996) to data quality improvement strategies, the so-called *Total Data Quality Management* (Wang 1998).

The essential problem in data quality management is the lack of precise definitions for data quality, thus making it difficult to (automatically) prevent or detect poor quality data. Apparently, this seems not to be a problem for data integrity where the *semantic correctness* of the data stored in a database

is specified by means of some logic-based formulas, e.g., the tuple relational calculus. Based on such specifications, integrity maintaining mechanisms such as triggers often can easily be derived (Grefen and Apers 1993, Ceri and Widom 1990).

In our opinion, the main difference between data quality and the traditional notion of data integrity is that data integrity concerns only the data stored in the database. In contrast, data quality additionally relates the data stored in the database with its “fitness for use”. That is, data quality also refers to applications and the portion of the real world modeled in the database. In fact, many important requirements we often refer to as data integrity cannot simply be specified by integrity constraints over a given database schema. For example, how to model (and maintain) the requirement that the data stored in a database are always complete with regard to the information we have in the real world? There exists no mechanism that forces the user, e.g., in a payroll office, to enter all information about all employees. In this case, traditional integrity constraints can only address the semantic correctness of the information stored about employees. Typical integrity constraints focusing on the completeness of information either require well defined attribute values for data records (thus preventing null values) or they describe foreign key conditions, thus assuming that already some portion of the relevant information has been stored.

Another prominent example often referred to as a data integrity problem is that a database should always contain only up-to-date data. But again, there neither exists a concept to formally specify this requirement nor to prevent storing outdated or expired data.*

Already these simple examples show that in order to ensure reliable data, we need new concepts and techniques that extend the traditional notion of data integrity and correctness. We cannot always expect high quality data or data that satisfy all integrity constraints. The new concepts must be rich enough to deal with less than total integrity as indicated in (Sheth 1997).

Although there does not exist a precise definition for data quality, the following dimensions are frequently used to analyze data quality aspects,* and will also be considered later in this paper.

1. *accuracy*, describing how precise and accurate real world information is mapped into local data structures (e.g., exact versus approximate values).
2. *completeness*, meaning that all real world information relevant for applications is recorded in the database.
3. *timeliness*, referring to the fact that the recorded information is up-to-date and not expired.

Note that in order to prevent incorrect data, these dimensions cannot be

*Unless one uses concepts that store the validity interval of data as it is done in temporal databases.

*In (Wang and Strong 1996) a survey is presented that lists 179 data quality attributes suggested by various data consumers.

specified formally for a database, and they are even more difficult to maintain. In a centralized database system the detection of data not satisfying some of the above dimensions often only occurs accidentally when, e.g., the retrieved data is compared with other information.

Beside the above rather intuitive notions, for our approach we furthermore add the dimension of *consistency* to data quality aspects. This aspect refers to the traditional notion of data integrity typically used in centralized database systems.

4. *consistency*, requiring that the data stored in a database satisfy the integrity constraints specified for the database.

Only for the dimension of data consistency data quality maintaining mechanisms exist. With regard to the other dimensions we have to cope with incorrect and poor quality data unless additional concepts, such as described in the following sections, allow us to identify and manage these data.

3.2 Data Quality in Data Integration

Recent reports on the usage of integrated data, in particular of data warehouses and decision support systems, show that the quality of integrated data causes major problems regarding the reliability of the information obtained through database federations. The reason for this is that quality and integrity of the data often has not been an issue for single, centralized database systems because users and applications “know” their data and they know how to (manually) cope with inconsistencies or rather poor quality data. Thus data having poor quality and integrity are often not detected until the usage of the data changes. This is exactly the scenario that happens when data are integrated from multiple database systems in order to fulfill the requirements of new, global applications.

In this section we present a concept that allows us to detect and handle poor quality data during data integration. The rationale behind this concept is that while it is difficult to detect incorrect or poor quality data in centralized databases, during data integration the quality of data can be compared. Such comparisons are typically performed when semantic data conflicts among the component databases containing semantically equivalent data are resolved. Existing approaches to data integration, as sketched in Section 2, handle such conflicts in a *static* manner. That is, a unique conflict resolution rule is given that specifies how data are integrated. However, as we will see below, for many such data conflicts there are no unique data integration rules because the correctness and quality of the data changes with time.

Example 1 Suppose two relations at two different component databases (CDBs) that store environmental data. Both relations (named `pollution`) contain data about the quantity of some toxic material recorded for different

regions. Assume that both relations have already been restructured and that they are “ready” to be integrated into a global relation `Pollution`.

Pollution@CDB1			Pollution@CDB2		
Region	Area	Quantity	Region	Area	Quantity
R1	A1	10	R1	A1	12
R1	A2	14	R1	A2	17
R1	A3	15	R1	A3	19
R1	A4	8	R1	A4	9
R2	B1	6	R2	B1	7
R2	B2	...	R2	B2	...
R2	...		R2	...	
R3	...		R3	...	
...			...		

In order to define a data integration rule for these two relations obviously a data conflict must be resolved due to the different quantities recorded for same regions. For this, data integration approaches suggest a conflict resolution function such that, e.g., in the global relation the quantity for an area is the average of the two values recorded for this area at CDB1 and CDB2. Now assume that for the above scenario we get further information from the local DBAs stating the following: CDB1 is updated on Mondays, Thursdays, and Saturdays, and CDB2 is updated on Tuesdays, Fridays, and Sundays. Thus there is no reason to miss out one relation for integration or to compute average data values.

In fact, both relations must be considered for integration, and the date a global query is issued determines from which relation to select up-to-date information. There is no unique integration rule ensuring the retrieval of up-to-date data because the data change with time.

Example 2 Suppose a third component database CDB3 with the following (restructured) relation:

Pollution@CDB3		
Region	Area	Quantity
R1	<i>null</i>	45
R2	<i>null</i>	...
...		

In comparison to the relations from CDB1 and CDB2, the third relation contains less accurate information. Therefore, data integration approaches would not consider this relation for integration. We argue that this relation still should be included, because in case both CDB1 and CDB2 are not reachable from the federation level, it is still possible to access information with less quality.

It should be mentioned that another reason for integrating all three relations (in a suitable manner) is that often users are interested in only “rough” information. If corresponding language constructs are provided within the global query language, the relation `Pollution@CDB3` would be sufficient to satisfy such requests. More importantly, some queries then can be processed much more efficiently because this relation already contains some aggregate data. Thus the data volume can be regarded as another useful data quality aspect.

The above examples all exhibit some kind of data conflicts, leading to the following observations:

1. Although the quality and integrity of the data at each individual component database can be high, concerning the integrated, global view of data, quality and integrity can be poor.
2. (Semantic) Data conflicts cannot always be resolved statically by specifying unique data integration rules at federation design time, because of the dynamic of the data stored at component databases. In such cases static rules would induce varying data quality and integrity at the integration level.
3. Integrating *poor quality data* or data with less integrity (as in the second example) sometimes is better than integrating no data at all.
4. Depending on the data requested by global applications, integrating poor quality data can decrease query processing cost.

Based on the above observations, another important question is how data integration rules should look like in the presence of data having (time-varying) quality. If all data have the same quality, of course, traditional integration rules can be adopted. For example, assuming that the first two relations in the example above contain data of the same quality, a global relation `pollution@GLOBAL` could simply be described by the data integration rule

$$\text{pollution@GLOBAL} := \text{pollution@CDB1}.$$

In case there are differences among the quality of data, roughly speaking, we have to integrate data from all three relations, i.e., we have the general integration rule `pollution@GLOBAL :=`

$$\text{pollution@CDB1} \cup \text{pollution@CDB2} \cup \text{pollution@CDB3}.$$

This is an essential difference to the traditional approaches to data integration where integration rules are based on whether the extensions of the relations to be integrated are disjoint or overlap. Furthermore, building the union of all relations, as suggested in our approach, requires a different approach for processing, more precisely decomposing, global queries that refer to `pollution@GLOBAL`. We discuss this aspect in more detail in Section 4.

Before we describe how different data quality aspects can be modeled at the integration level, we discuss “suitable” strategies handling the integration of integrity constraints that may exist at component databases. Despite detailed

proposal on how to handle integrity constraints in database integration (e.g., (Reddy *et al.* 1995, Vermeer and Apers 1996, Conrad *et al.* 1997)), there is no consensus on an optimal strategy for designing or deriving global integrity constraints. The two extremes range from not to consider integrity constraints at all (often suitable for global read-only applications) to the integration of the most restrictive integrity constraints.

We argue that often none of the proposed approaches fulfill practical needs to cope with less than absolute integrity. For example, not considering some information from one component database because it does not satisfy a global integrity constraints prevents the integration from perhaps “useful” information. If, for instance, some tuples from a relation stored at a CDB do not satisfy a simple global domain constraint, the tuples still may carry some useful information and should be considered for integration. This, however, requires that we suitably model this aspect at the integration level. The aspect of explicit handling integrity constraints in global query processing and their usage for “data scrubbing” is discussed in Section 4.

3.3 Modeling Data Quality

In order to suitably model data quality and integrity aspects discovered during conflict resolution and data integration, additional constructs must be provided at the integration level, more precisely, within the metadata repository. These additional metadata record respective information and make this information accessible to global applications and to the query processor. Although it seems that data quality and integrity aspects are just some further attributes that can be associated with component databases and stored information, it is difficult to simply assign, e.g., discrete values to sites and information such as “complete”, “less complete”, or “not complete”.

As shown in the previous section, quality aspects can often only be detected by comparing information from two CDBs. In order to model these aspects, we define a set $\mathcal{Q} := \{Q_1, \dots, Q_6\}$ of *basic quality dimension*:

Q_1	accuracy	Q_4	availability
Q_2	completeness	Q_5	reliability
Q_3	timeliness	Q_6	data volume

Note that with regard to the quality aspects listed in Section 3.2, we have added the dimensions Q_4 , Q_5 , and Q_6 which are only relevant for information accessed through a federation. We suggest that the dimension *reliability* is used only if no other dimensions such as accuracy, completeness, or timeliness can be applied. In this respect, reliability can be considered as a generalization of Q_1 , Q_2 , and Q_3 . Further dimensions, of course, are possible and should be included depending on the properties and usage of integrated data.

Comparisons of data quality can occur at different levels of granularity, e.g., we can compare relations at different CDBs or CDBs as a whole. We assume that data quality comparisons with regard to a quality dimension

Q_i must occur on *information units* of the same granularity. The different *types of granularity* we consider are component databases (C), relations (R), (selected) tuples from a relation (determined through a selection $\sigma_{\mathbf{F}}(\mathbf{R})$) (T), and projections on attributes of (selected) tuples (A).

For the different types, we assume a function *origin* that can be applied to any information unit and returns the name of the CDB the unit is contained in. For example $origin(CDB) = CDB$, or $origin(R@CDB1) = CDB1$. As we will show later, this function can be useful to query the origin of poor quality data retrieved by a global query. Comparisons of data quality between semantically related information units having the same granularity are defined as follows:

Definition 3 Let $gran(I)$ denote the granularity of an information unit, i.e., $gran(I) \in \{C, R, T, A\}$. For two semantically related information units I_k, I_l such that $gran(I_k) = gran(I_l)$, a data quality comparison with regard to a quality dimension $Q_i \in Q$ is denoted by $I_l \Theta_{Q_i} I_k$, $\Theta \in \{>, =\}$.

The meaning of $I_l \Theta_{Q_i} I_k$, also called *quality statement*, is that the quality of the information represented by I_l is higher ($\Theta \equiv >$) or equal ($\Theta \equiv =$) to the quality of the information represented by I_k wrt the quality dimension Q_i .

Data quality comparisons among semantically related information units, say two relations R_1 and R_2 which share an attribute A , can be specified formally. R_1 is said to be more *up-to-date* (at time point t) than R_2 iff R_1 contains more tuples than R_2 that have recently been updated on A .

It is important to note that the knowledge necessary to determine quality statements can often be derived from *information profiles*. Such profiles, which can be associated with information units of any granularity, describe the information processing techniques employed to map real world data into local data structures and the maintenance of these data.

The coarsest information unit quality comparison can be applied to are CDBs, the finest granularity is determined by attributes of (selected) tuples stored in a relation at a CDB. For tuples and attributes describing I_l and I_k , we require that the same selection condition \mathbf{F} and projection is used.

Since one cannot expect that for all information units and all quality dimensions quality aspects are known and thus can be compared, we take the following assumptions.

Assumptions 4

- A1** For a quality statement $I_l \Theta_{Q_i} I_k$ with $gran(I_l) = gran(I_k)$, we assume that for all information units I'_l, I'_k of finer granularity with $origin(I'_l) = origin(I_l)$ and $origin(I'_k) = origin(I_k)$ the quality statement $I'_l \Theta_{Q_i} I'_k$ holds.
- A2** Exceptions of assumption **A1** must be specified explicitly, i.e., for $I_l \Theta_{Q_i} I_k$ we can have a quality statement $I'_k \Theta_{Q_i} I'_l$.
- A3** For all semantically related information units from the different CDBs having the same granularity, we assume that if no quality statement with

regard to Q_i has been specified, induced by **A1**, or no exceptions are given, then these information units all have the same quality wrt Q_i (which can be different from those specified explicitly with regard to Q_i).

It should be mentioned that there are many special cases for the different quality attributes. For example, the quality dimension Q_4 (availability) can only be associated with the coarsest information units, i.e., component databases. Assumption **A1** is useful in order to specify, e.g., that all data stored in one component database are in general more complete than the data of another CDB. This property then is automatically inherited to the relations at these CDBs. For certain relations at these CDBs, however, we allow exceptions which must be specified explicitly.

A set \mathcal{S} of quality statements over a set \mathcal{C} of component databases and associated information units of finer granularity has to satisfy certain properties.

Definition 5 A set \mathcal{S} of quality statements is *correct* iff there exists no contradiction between explicitly specified quality statements, and \mathcal{S} is *complete* iff every information unit is considered in an explicit or implicit quality statement.

Although assumption **A3** ensures the completeness of quality statements, it does not state how to handle information units that all have the same quality. Assume that we have five component databases, and each CDB contains a relation P to be integrated into a global relation P . Furthermore suppose that the only quality statements we have are $P@CDB1 >_{q_2} P@CDB2$ and $P@CDB1 >_{q_2} P@CDB3$. Due to **A3**, we have that $P@CDB2 =_{q_2} P@CDB3$, but we also have that $P@CDB4 =_{q_2} P@CDB5$. But how are these two relations related to the relations at CDB1, CDB2, and CDB3 with regard to Q_2 ? One could require that the designer of the federation has to solve such cases by connecting $P@CDB4$ and $P@CDB5$ with at most one of the other three relations by quality statements. A more appropriate approach, however, would be to consider $P@CDB4$ and $P@CDB5$ as a separate class that must be handled appropriately by the query processor (see below).

As discussed earlier and shown in Example 1, the quality of data may change as time advances. Above quality statements, however, only describe static properties of data quality. For this reason, we allow that with each quality statement a temporal condition TC can be associated. A temporal condition essentially describes a validity interval that specifies *when* a quality statement $I_l \Theta_{Q_i} I_k$ holds. For the sake of simplicity, we assume that a validity interval either can be described by a start and end date, e.g., [01-01-96, 12-31-97], or by a set of explicit dates. If no temporal condition is specified, we assume that $I_l \Theta_{Q_i} I_k$ always holds. More useful and complex specifications for validity intervals, of course, should be investigated. In this context it might even be useful to associate a time interval with a single information unit, leading to a similar concept as it can be found in temporal databases. Including temporal

conditions also requires extending the notion of a complete and correct set of quality statements which, however, is trivial and will not be discussed here.

Example 6 Assume the three relations from the CDBs discussed in Example 2. The specification of the quality statements could look as follows (with $R \equiv \text{Pollution}$):

$$\begin{aligned}
 Q_1 : & \quad R@CDB1 =_{q_1} R@CDB2 >_{q_1} R@CDB3 && \text{(Accuracy)} \\
 Q_2 : & \quad R@CDB1 =_{q_2} R@CDB2 >_{q_2} R@CDB3 && \text{(Completeness)} \\
 Q_3 : & \quad R@CDB1 >_{q_3} R@CDB2 \text{ (Timeliness)} \\
 & \quad \text{with } TC = \text{sysdate.day} \in \{ \text{'monday'}, \text{'thursday'}, \text{'saturday'} \} \\
 & \quad R@CDB2 >_{q_3} R@CDB1 \text{ (Timeliness)} \\
 & \quad \text{with } TC = \text{sysdate.day} \in \{ \text{'tuesday'}, \text{'friday'}, \text{'sunday'} \} \\
 Q_4 : & \quad CDB1 =_{q_4} CDB2 >_{q_4} CDB3 && \text{(Availability)} \\
 Q_6 : & \quad R@CDB1 =_{q_6} R@CDB2 >_{q_6} R@CDB3 && \text{(Data volume)}
 \end{aligned}$$

Although we have not given a complete formal specification of the language that can be used to formulate quality statements and temporal conditions, it should be obvious that a complete formalism can easily be developed and that respective specifications can be represented in the metadata repository. Interestingly, correctness and completeness of quality statements then can be considered as integrity constraints imposed on respective metadata.

The final step in modeling data quality aspects consists of handling local integrity constraints, which are specified in the global data model. For this, we take a rather pragmatic approach which, we think, is most appropriate if global applications only access integrated data but do not modify these data. We assume that for each component database CDB_i a set of local integrity constraints is specified. In practice, these constraints turn out to be quite simple, mainly restricted to domain and foreign key constraints. The definition of each constraint and the names of the global relations the constraint affects is recorded in the metadata repository. Information about integrity constraints that cannot be associated with a component database but which have been derived in combination with conflict resolving rules are recorded as well.

The discussions in this section show that the metadata repository plays an important role during the database integration task. All information related to data quality and data integrity aspects are recorded as metadata in addition to conflict resolving and data integration rules.

4 QUERY PROCESSING

In the federated database approach, the global schema is employed to submit a global query. Objects of the global schema are transparent to the user, i.e., the user is unaware of the component databases and the data integration rules encoded in the metadata. The metadata are used by the global query processor which decomposes the global query into global subqueries such that the

data needed by each subquery are available from one CDB. The global query processor also translates each global subquery into queries of the corresponding CDB and finally combines the results returned by the subqueries. For a detailed discussion about query processing in federated and multidatabase system, we refer the reader to, e.g., (Meng and Yu 1995) or (Evrendilek *et al.* 1997).

In the presence of different data quality aspects, which are encoded in the metadata, it is quite obvious that global queries cannot simply be decomposed solely based on the prespecified unique data integration rules. Assume, for example, the global relation `Pollution[@GLOBAL]` defined by the integration rule `pollution := pollution@CDB1 \cup pollution@CDB2 \cup pollution@CDB3` (see also Example 2) and that we want to issue the query

```
select Region, sum(Quantity) from Pollution[@GLOBAL]
group by Region
```

Using only the above integration rule we would get an erroneous result. But what do we expect? The general strategy should be that we always want high quality data. That is, data integration rules should exclude (time-varying) poor quality data. However, provided that we have suitable language constructs in our global query language, one could also be interested in *all data* which include data of different quality. In this case we get a *multiresolution query*. For the above example, we get three sets of tuples, and each set must suitably be represented to reflect differences in the quality of the result (in the above case outdated and less accurate data must be indicated).

Given a set of quality statements including temporal conditions, the problem of global query processing can be reduced to the following issues:

1. Tuples and attributes having different quality cannot simply be combined using the union operator, and
2. tuples having different quality cannot be joined, e.g., it must not be possible to join outdated or expired data (tuples) with up-to-date data.

In this paper we suggest a preliminary concept of a conservative approach to these problems. For this, we assume that the main usage of the integrated data is not based on ad-hoc queries, but applications having well-defined quality requirements with regard to the data to be retrieved. We also assume that typical global queries are simple and do not contain complex subqueries. Different global applications may have different requirements concerning the quality of integrated data used in the applications. Rather than to encode these requirements explicitly in numerous data integration rules tailored to certain applications, global queries on global relations can be enriched by data quality conditions. These conditions are specified by the designer who employs the quality statements encoded in the metadata repository in a semi-transparent fashion. For this, we suggest an extension of the global query language, say

SQL, by quality predicates. These predicates specify the *data quality goal* of a global query as shown in the general pattern

```

select <list of attributes>
from <list  $R_1, \dots, R_n$  of global relations>
where <SQL selection condition>
with goal <list of data quality goals>;

```

The first goal in the list of data quality goals designates the *primary data quality goal*, the next goal the secondary data quality goal and so on. A simple goal is either *most up-to-date*, *most accurate*, *most complete*, or *most reliable*.

Consider, for example, the primary goal *most up-to-date*, which refers to the timeliness of the integrated data. In this case, for each global relation R_i , the query processor reduces the corresponding general data integration rule to an integration rule that considers only the most up-to-date local relations that build up R_i at the time point t where there query is issued. Information about the most up-to-date local relations can easily be determined by using the quality statements and associated temporal conditions. If several local relations satisfy the specified goal, only those relations are chosen that satisfy the secondary goal best. If we allow temporal conditions for quality statements, only those information units (relations) are considered in the integration rule that satisfy these conditions. Thus data quality specific data integration rules are generated dynamically from general integration rules that do not consider data quality aspects.

Example 7 Assume a global query referring to global relations $R (= R_1 \cup R_2)$ and $S (= S_1 \cup S_2 \cup S_3)$ and that we have the quality statements $R_1 >_{q_1} R_2$ and $S_1 >_{q_1} S_2 =_{q_1} S_3$ referring to the accuracy of the integrated data. With the data quality goal *most accurate* the query processor reduces the two general data integration rules to $R = R_1$ and $S = S_1$. In case there are no quality statements about the accuracy of the local relations R_1, R_2 , the original data integration rules are chosen for R and S (which include duplicate elimination).

Query language constructs to retrieve data not satisfying the specified query goal(s) must be provided as well in order to compare data of different quality and thus to determine poor quality data (and its origin). Assume a global relation S as specified above, but with the quality statements referring to the completeness. The information that this relation consists of two groups of data having different quality can easily be represented to the designer. The global query language furthermore must provide the designer language constructs to retrieve all data or only data that belong to one of these two groups. Provided that respective language constructs exist, the designer can retrieve groups separately, and he can specify select statements that retrieve tuples which occur in all or only specified groups. The basic idea behind this concept is that the designer has the possibility to compare quality aspects of integrated data at the federation level. The information obtained about these

aspects then leads to the formulation of global queries that are used by global applications.

Before we conclude this paper in the next section, we finally discuss how integrity constraints should be considered at the integration level and for global applications. We suggest to employ local integrity constraints (formulated in the global data model) to *filter* tuples retrieved by global queries. Assuming that the constraints specified at CDBs are simple, restricted, e.g., to domain and foreign key constraints, sophisticated techniques for *data scrubbing* can be developed.

Assume, for example, the designer wants to retrieve data from a global relation R (for which quality statements exist). Provided that there are suitable tools at the federation layer, information about the existence of integrity constraints affecting this relation can be displayed to the designer. Recall that the information about integrity constraints is recorded in the metadata and thus needs only be represented suitably. The designer then can select some of these constraints and can impose these constraints to his original query. The query result still would be the same, but result tuples violating the selected constraints are highlighted by, e.g., a certain coloring schema. The query result then can be reduced to those tuples from R that violate selected constraints and the designer then can investigate the *origin* of these tuples. For this, the query processor can provide information about the integration of data from local relations that build up the global relation R . Knowing the origin of tuples that violate integrity constraints can lead to improvements of data quality and integrity at local component databases.

The above examples and discussions show that many sophisticated global query processing concepts are possible from which we have sketched only a very few. A development of a complete query language with a precise semantics as well as associated efficient query processing techniques are subject to future research. However, the main idea of data quality based query processing as outlined above is quite obvious: data quality aspects and their consideration in query formulation and query processing is *semi-transparent* to the designer. That is, the designer is not responsible for reducing general data integration rules to certain unions of local relations. However, the designer is aware of the fact that a global relation contains data of different quality. He can use this information (provided by the query interface) to build global views that retrieve only data satisfying certain quality goals. These views then can finally be associated with different global applications. For these applications, the existence of data having different quality then is totally transparent. Depending on the type of applications, however, it is still possible to represent different data quality and integrity aspects in a suitable fashion.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have outlined a framework for modeling and managing data quality and integrity aspects in database integration. In particular, we have motivated that existing approaches to data integration typically fail to address these issues, because of the assumption that conflict resolution and data integration rules are of static nature. As shown by means of several examples, however, aspects such as outdated or incomplete data are often rather of dynamic nature. That is why we have introduced a taxonomy of data quality and integrity aspects that can be used to specify (time-varying) statements about data quality among components databases and relations at federation design time. The respective information, which is stored in the federation's metadata repository, then can be (1) employed by the designer to specify data quality goals for global queries and (2) used by the query processor to dynamically build data integration rules from general ones that are tailored to the specified data quality goals.

In our future research we want to address the following issues:

- A complete formal specification of data quality statements and an (incremental) modeling approach. The latter issue is important because we cannot expect a designer to detect and specify all data quality aspects at federation design time. It seems more appropriate to investigate deficiencies in the data quality reported by global users and to record corresponding information in the metadata repository. This would lead to an incremental data quality improvement strategy.
- An extension of a global query language, such as SQL, which allows users and designers not only to formulate different query goals, but also to represent retrieved data having different quality in a suitable manner. That is, representation techniques for multiresolution queries need to be developed. The extension of a global query language includes a well-defined syntax and semantics of data quality and data integrity related language constructs.
- Efficient query processing techniques for tailoring general data integration rules to rules which integrate only data satisfying data quality goal(s) that are specified in the global query.

In conclusion, we are convinced that the role of data quality in database integration (in addition to data integrity) needs much more attention in order to ensure correct and meaningful integrated data. The “traditional” assumption that data conflicts such as outdated data, incorrect data etc. among multiple information sources can always be resolved by unique data integration rules mainly fails. The main reason for this is that in today's database applications data typically change with time and thus inconsistencies and data quality properties are of dynamic nature.

REFERENCES

- J. Bischoff, T. Alexander: *Data Warehouses: Practical Advice from the Experts*. Prentice-Hall, 1997.
- M. Bright, A. Hurson, S. Pakzad: A Taxonomy and Current Issues in Multi-database Systems. *IEEE Computer* 25:3 (March 1992), 50–59.
- C. Batini, M. Lenzerini, S. B. Navathe: A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys* 18:4 (December 1986), 323–364.
- Y. Breitbart, H. Garica-Molina, A. Silberschatz: Transaction Management in Multidatabase Systems. In (Kim 1995), 573–591.
- S. Conrad, M. Höding, G. Saake, I. Schmitt, C. Türker: Schema Integration with Integrity Constraints. In C. Small, P. Douglas, R. Johnson, P. King, N. Martin (eds.), *Advances in Databases, 15th British National Conf. on Databases, BNCOD 15*, 200–214, Lecture Notes in Computer Science 1271, Springer-Verlag, Berlin, 1997.
- S. Ceri, J. Widom: Deriving Production Rules for Constraint Maintenance. In McLeod, D., Sacks-Davis, R., and Schek, H. (eds.), *Proceedings of the 16th International Conference on Very Large Data Bases - 1990*, 566–577, Morgan Kaufmann Publishers, 1990.
- C. Evrendilek, A. Dogac, S. Nural, F. Ozcan: Multidatabase Query Optimization. *Distributed and Parallel Databases* 5 (1997), 77–114.
- P. W. Grefen, P. M. Apers: Integrity Control in Relational Database Systems – An Overview. *Data & Knowledge Engineering*, 10(2):187–223, 1993.
- M. Jarke, Y. Vassiliou: Data Warehouse Quality Design: A Review of the DWQ Project. Invited Paper, *Proc. of the 2nd Conference on Information Quality*. Massachusetts Institute of Technology, Cambridge, 1997.
- W. Kim: *Modern Database Systems: The Object Model, Interoperability, and Beyond*, ACM Press, New York, 1995.
- R. Kimball: Dealing with Dirty Data. *DBMS Magazine* 9:10, September 1996, Miller Freeman, Inc, 1996.
- W. Kim, I. Choi, S. Gala, M. Scheevel: On Resolving Schematic Heterogeneity in Multidatabase Systems. In (Kim 1995), 521–550.
- W. Kim, J. Seo: Classifying Schematic and Data Heterogeneity in Multi-database Systems. *IEEE Computer* 24:12 (December 1991), 12–18.
- W. Litwin, L. Mark, N. Roussopoulos: Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys* 22:3 (1990), 267–293.
- W. Meng, C. Yu: Query Processing in Heterogeneous Environment. In (Kim 1995), 551–572.
- M. Rusinkiewicz, P. Missier: Extending a Multidatabase Manipulation Language to Resolve Schema and Data Conflicts. In R. Meersman, L. Mark (eds.), *Database Applications Semantics, Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)*, 93–115, Chapman & Hall, London, 1995.

- T.C. Redman: *Data Quality for the Information Age*. Artech House, Boston, MA, 1996.
- M.P. Reddy, B.E. Prasad, A. Gupta: Formulating Global Integrity Constraints During Derivation of Global Schema. *Data & Knowledge Engineering* 16:3 (1995), 241–268.
- A. Sheth: Pragmatics Driven Research Issues in Data and Process Integrity in Enterprises. In L. Strous, S. Jajodia, G. McGregor, W. List (eds.), *Integrity and Control in Information Systems*, Chapman & Hall, London, 1997.
- A. Sheth, J. Larson: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys* 22:3 (1990), 183–236.
- A. Sheth, V. Kashyap: So Far (Schematically) Yet. So Near (Semantically). In D. Hsiao, E. Neuhold, R. Sacks-Davis (eds.), *Interoperable Database Systems (DS-5)*, North-Holland, Amsterdam, The Netherlands, 1993.
- S. Spaccapietra, C. Parent, Y. Dupont: Model Independent Assertions for Integration of Heterogeneous Schemas. *VLDB Journal* 1:1, 81–126, 1992.
- M. W. Vermeer, P. M. Apers: The Role of Integrity Constraints in Database Interoperation. In T. Vijayaraman, A. Buchmann, C. Mohan, N. Sarda (eds.), *VLDB'96 – Proceedings of the 22th International Conference on Very Large Data Bases, Sept. 3-6, 1996, Bombay*, 425–435, Morgan Kaufmann Publishers, 1996.
- R.Y. Wang: A Product Perspective on Total Data Quality Management. *Communications of the ACM* 41:2, 58–65, 1998.
- R.Y. Wang, D.M. Strong: Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems* 12:4, 5–34, 1996.
- R.Y. Wang, V.C. Storey, Firth, C.P.: A Framework for Analysis of Data Quality Research. *IEEE Transactions on Knowledge and Data Engineering* 7:4 (August 1995), 623–640, 1996.

BIOGRAPHY

Michael Gertz is an Assistant Professor of Computer Science at the University of California at Davis. He received his diploma degree in Computer Science from the University of Dortmund, Germany, in 1991, and the Ph.D. degree in Computer Science from the University of Hannover, Germany, in 1996. His research interests are in the design and implementation of multidatabase systems. His current research is on database integration techniques, database interoperability, constraint maintenance in multidatabase systems and integrity enforcement in temporal databases. His research interests also include the design of user interfaces to database systems and database administration tools.