Data Integration Techniques based on Data Quality Aspects

Michael Gertz

Department of Computer Science

University of California, Davis

One Shields Avenue

Davis, CA 95616, USA

gertz@cs.ucdavis.edu

Ingo Schmitt

Institut für Techn. Informationssysteme

Otto-von-Guericke-Universität Magdeburg

Universitätsplatz 2

D-39106 Magdeburg, Germany

schmitt@iti.cs.uni-magdeburg.de

Abstract

In multidatabase systems, a major data integration problem is to resolve data conflicts where two objects having the same definition and representing the same real world object have different extensions. Traditional data integration approaches suggest static conflict resolution functions that perform a computation over conflicting attribute values, thus assuming a unique and time-independent resolution.

In this paper, we argue that such type of data conflict often arises due to heterogeneities among the data capturing and processing techniques and methods used by component databases. That is, component database differ in how and when they map real world data into local data structures. This diversity results in the fact that the quality of the data stored at different sites can be different and that the quality can also vary over time, thus requiring dynamic data integration methods, depending on which data quality goal is required at the global level. We outline a novel framework that allows to formalize, model and utilize diverse and in particular orthogonal data quality aspects such as timeliness, accuracy, and completeness in database integration. By making the notion of data quality aspects explicit both in modeling and querying a multidatabase system, existing approaches to database integration cannot only be extended, but also tools can developed that ensure different types of "high quality data" at the integration level and for global applications.

1 Introduction

The integration of previously independent but semantically related database systems into globally accessible multidatabase systems (MDBS) has been one of the most active areas in database research over the past 15 years. Most of the work has been devoted to approaches and techniques that allow designers to identify and resolve structural and semantic conflicts between meta-data objects (tables, classes, ...) and data items (records, tuples, objects, ...) located at local databases participating in the multidatabase system [BE96, KCG95, KS91, She91]. Despite the known difficulties in detecting semantic equivalences and resolving semantic conflicts and heterogeneities, many products have emerged that realize some kind of a multidatabase system.

The most prominent type of such systems are data warehouses [Kim96, BA97]. Data warehouses typically restrict access to local databases to read-only accesses, providing global users and applications a means to integrate data from different resources for, e.g., decision support systems, hospital information systems or environmental information systems. Although the issue we discuss in this paper is not specific to data warehouses but any system that provides access to integrated data, most of the problems related to integrated data have been reported for data warehouses. As recent studies and reports show, in particular applications build on top of data warehouses often experience several problems with regard to the reliability and quality of the integrated data [Kim96b, Huf96, BA97, JV97]. The main reason for this is that often already the local databases participating in the multidatabase system contain incorrect, inaccurate, outdated or simply poor quality data. The quality of integrated data then becomes even worse unless suitable methods and techniques for modeling and managing these issues are employed during multidatabase design time. Despite the amount of work that focuses on semantic heterogeneity among meta-data and data items, aspects of the quality of integrated data have not been addressed so far. Data quality mainly has been and still is an important research topic independent of database integration [Red96, Wan98, WS96, WSF96].

In this paper we explicitly introduce the notion of data quality in database and data integration. We claim that often data conflicts between local data items occur due to discrepancies among how the data are gathered, processed and maintained at local databases. This type of heterogeneity, called *operational heterogeneity*, then can result in the fact that the *quality* of information stored about real world objects or artifacts in different local databases may be different and vary over time. We show that data quality can be defined as orthogonal aspects, referring to timeliness, accuracy and completeness of data. The orthogonality of these aspects indicates that there is not always a unique resolution of data conflicts. For example, if the attribute values of two objects referring to the same real world object differ and one value is know to be more accurate and the other is known to be more up-to-date, one cannot give a unique resolution (or data integration rule) for this conflict. More importantly, accurate data does not necessarily imply up-to-date data or vice versa. Therefore, in this paper we suggest some kind of measurement for different data quality aspects. These measurements are used to model data quality aspects of local and global classes at multidatabase design time. We present a technique called *information profiling* that allows designers to associate and relate data quality aspects with different portions of semantically equivalent classes for which data conflicts can occur.

Information about data quality measurements as well as preferences among local objects and attribute values with respect to different data quality aspects are recorded as metadata at the integration level. The transparency of global query processing is weakened by that a global user or application can specify a data quality goal for the result of a global query. Based on such data quality goals and the recorded metadata, data integration rules are generated dynamically by the global query processor to ensure the retrieval of "high quality data" from local databases.

A typical data integration scenario which reveals the different aspects of data quality and also the need for dynamic data integration rules is discussed in the next section. Throughout the paper we adopt a simple object-oriented data model, not focusing on aspects like complex objects, methods or object/method inheritance.

1.1 Motivating Example

Suppose two classes at two local databases DB_1 and DB_2 which store environmental data. Both classes (named `Pollution`) contain data about the quantity of some toxic materials `Mat1` and `Mat2` recorded for different regions. We assume that schematic conflicts between the two classes have been resolved and now a data integration rule for the global class, say `G_pollution`, needs to be specified. Below are the extensions of the two classes:

Pollution@DB1 (C_1)

Region	Area	Mat1	Mat2
R1	A1	10	2
R1	A2	14	3
R1	A3	15	3
R1	A4	8	1
R2	B1	6	1
R2	B2	7	3
R4	A1	21	8

Pollution@DB2 (C_2)

Region	Area	Mat1	Mat2
R1	A1	12	2
R1	A2	17	2
R1	A3	19	3
R2	B1	7	2
R2	B2	7	1

In order to define an appropriate data integration rule for these two classes, obviously a data conflict has to be resolved due to the different quantities recorded for same regions. Traditional data integration approaches suggest a conflict resolution function that either chooses one class (or attribute) over the other or that computes the average of the values recorded for the same region. A global query against `G_Pollution` then always retrieves the data from the two class extensions according to the specified data integration rule. Now assume the following scenarios where additional information about the two classes and their extensions has been obtained:

Scenario 1: DB_1 updates its class C_1 on Mondays, Thursdays, and Saturdays, and DB_2 updates its class C_2 on Tuesdays, Fridays, and Sundays. In this case, the time a global query is issued against the global class `G_Pollution` determines from which extension *up-to-date* data are retrieved.

Scenario 2: While the quantity of `Mat1` in C_1 is recorded using some kind of sensors, the values for `Mat1` in C_2 are recorded on a manual basis. Thus the values for `Mat1`.C_1 may be more *accurate* then the values recorded in `Mat1`.C_2.

Scenario 3: DB_1 covers more regions in C_1 than DB_2 does in C_2. That is, the information in C_2 is less *complete* than the information in C_1.

The above three simple scenarios, which will be formalized in the next section, show that the way of how and when data is populated into local classes plays an important role in integrating local data for global queries. Interestingly, the above scenarios also describe orthogonal aspects of data quality. That is, for example, up-to-date data does neither necessarily imply most accurate nor most complete data. More importantly, while one global user (or application) might be interested in most complete data, another user might be interested in most accurate data. In both cases there must be the possibility for a user to specify certain data quality goals or, at least, the global query interface should reflect that the data retrieved from different sources in response to a global query have different quality, e.g., trough tagging or grouping retrieved records.

Furthermore, the above data quality aspects are not only of interest in case of overlapping (and conflicting) extensions of local classes, but also if semantically equivalent classes are

disjoint. Thus the above scenarios describe more than just data conflicts.

1.2   Comparison with other Work

The methods and techniques described in this paper are influenced by two areas: database integration and data quality management. In the area of database integration most of the work focus on resolving structural conflicts on the schema level (schematic conflicts) including class and attribute conflicts, e.g., [BLN86, SPD92, GSC96, KCG95]. Less work has been done on detecting and resolving so-called semantic data conflicts and semantic heterogeneities at the data level, typically because it is difficult to formalize and compare semantics associated with data items and data values. Classifications and discussions on semantic issues in multidatabase systems can be found in [She91, SK93, KS96] and in particular [BE96]. Special types of semantic conflicts, in particular the resolution of different object identifiers, have been discussed in [CTK96, Pu91, SS95, DeM89]. Recent and very promising work addressing semantic issues in database interoperability focus on context-based approaches, e.g., [GMS94, KS96]. Semantic conflicts of types like inconsistent data or outdated data have been listed in some work (e.g., [SK93, BE96]), but their identification and resolution has not been discussed in detail.

In the area of data quality management there has been no particular focus on integrated data or multidatabase systems in general. Most of the work focuses on definitions and measurements of data quality aspects in database and information systems [Red96, Wan98, WS96, WSF96]. Only [RW95] discusses the estimation of the quality of data in federated database systems, thus not considering the integration aspect or task but only the already integrated data. In [JV97, JJQ98] the aspect of data quality is addressed in the context of data warehouses but neither formal definitions or measurements for data quality nor design methodologies focusing on different data quality aspects are given.

1.3   Paper Outline

In Section 2 we show how the data quality aspects timeliness, accuracy and completeness can be formally defined using the notion of virtual classes. Because virtual classes and their properties are not directly accessible at multidatabase design time, in Section 3 we present an approach called information profiling that allows to identify and compare different data quality aspects associated with local classes and their possible extensions. The result obtained through information profiling is recorded as metadata at the integration layer and is used for global query processing, which is outlined in Section 4.

2   Formalization of Data Quality in MDBS

The quality of data stored at local databases participating in a multidatabase system typically cannot be described or modeled by using discrete values such as "good", "poor" or "bad". In this section, we suggest a specification of (time-varying) data quality assertions based on comparisons of semantically related classes and class extensions. In Sections 2.1 and 2.2 we give formal definitions of different data quality aspects using virtual classes.

2.1 Virtual and Conceptual Classes

In order to analyze and determine the conflicts or semantic proximity of two objects, object attributes or even complete classes, one needs to have some kind of a reference point for comparisons. In this paper, such comparisons are based on *virtual classes*. A virtual class is a description of real world objects or artifacts that all have the same - not necessarily completely instantiated - attributes. The extension of a virtual class is assumed to be always up-to-date, complete and correct, i.e., only current real world objects and data are reflected in the extension of a virtual class.

Assuming such type of classes is quite natural in database design and they are also used to describe semantic issues in database interoperability, e.g., in [SG93, GSC96]. The reason for this is that the development of local database schemas is typically driven by what information about real world objects and classes is needed for local applications and what information is available about these classes and objects.

Given a description of real world objects in terms of a virtual class C_{virt}, local databases $DB_1, \ldots, DB_m$ typically employ only a partial mapping of real world data into local data structures, i.e., only information relevant to local applications is mapped into local classes $C_1, \ldots, C_n$. More importantly, the mappings $\alpha_1, \ldots, \alpha_n$ adopted by local databases differ in the underlying local data structures (schemata) and how real world data is populated into these structures. Different mappings then result in schematic and semantic heterogeneities among the local classes $C_1, \ldots, C_n$ that refer to the same virtual class C_{virt}. While a local class C_i typically maps only a portion of the information associated with C_{virt}, a *conceptual (or global) class* C_{con} integrates all aspects modeled in semantically equivalent or similar local classes (Figure 1).
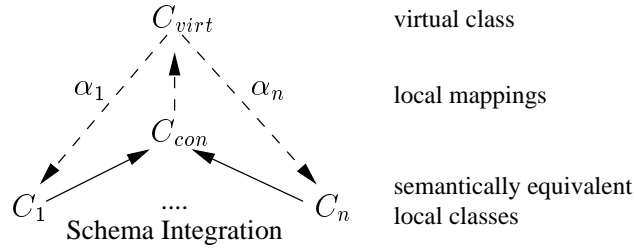

Figure 1: Relationship between virtual, conceptual and local classes

Determining conceptual classes as components of, e.g., a federated or multidatabase schema, is the main task in database integration. One main goal in database integration is that the specification of a conceptual class C_{con} *comes as near as possible* to the specification of the associated virtual class C_{virt} from which the local classes $C_1, \ldots, C_n$ are derived. Beside these structural aspects of conceptual classes, the other aspect is how to integrate data from (now structurally equivalent) local classes into global classes. That is, data integration rules need to be specified. If the (possible) extensions of local classes are known to be disjoint, then the data integration rule basically consists of joining the extensions of these local classes. In case of possible object or data conflicts, that is, for an object of the global class there are at least two local objects corresponding to that object but with different attribute values, data integration rules additionally describe conflict resolution. Such resolutions ensure that only one object is retrieved from local classes or local attribute values from same objects are combined appropriately into one global object.

5

2.2 Basic Data Quality Aspects

A basic property of real world objects is that objects as instances of virtual classes evolve over time. Objects are added and deleted, or properties of objects change. At local sites different organizational activities are performed to map such time-varying information into local classes, thus resulting in a type of heterogeneity among local databases we call *operational heterogeneity*. We consider operational heterogeneity as a non-uniformity in the frequency, processes, and techniques by which real world information is populated into local data structures. For example, while at one local database the stored data are updated manually on a weekly basis, at another database storing related information updates are performed automatically (e.g., sensor-based) on a monthly basis. In such cases, operational heterogeneity can lead to the fact that similar data referring to same properties and attributes of real world objects have different *quality* and thus may have varying reliability. Based on the used data capturing and processing approaches, the quality of data at a local database can even vary over time, for example, data can be outdated, as discussed in Section 1.1. More importantly, we have also shown that operational heterogeneity cannot be resolved on the schema integration level but requires a suitable approach to data integration.

These observations raise the question whether existing approaches to resolving semantic data conflicts are appropriate or rich enough to handle aspects such as outdated data or wrong data. In this paper we consider operational heterogeneity and data quality in particular as a type of semantic data conflict which requires a not necessarily unique resolution at run-time by means of (time-varying) data integration rules.

We first give a formal definition of time-varying data quality aspects we consider as most important in database integration and resolving semantic data conflicts. For this we make the following (simplified) assumptions.

- There are two classes $C_1\langle A_1, \dots, A_n \rangle$ and $C_2\langle A_1, \dots, A_n \rangle$ from two local databases DB_1 and DB_2. C_1 and C_2 refer to the same virtual class C_{virt} and schema integration has been performed for C_1 and C_2 into the conceptual class $C_{con}\langle A_1, \dots, A_n \rangle$. We assume that the two classes are represented in the global data model. The resolution of heterogeneous representations of the original local class structures is a topic of schema integration and is therefore outside the scope in this paper.

- Using the predicate *same* it is possible to determine whether an object o_1 from the extension of C_1, denoted by $Ext(C_1)$, refers to the same real world object $o \in Ext(C_{virt})$ as an object $o_2 \in Ext(C_2)$. The resolution of differences among key representations is assumed to be resolved during schema integration by appropriate methods as suggested in, e.g., [Pu91, CTK96, SS95].

As motived in Section 1.1, the aspect of time plays an important role in operational heterogeneity. For this, we assume a discrete model of time isomorphic to natural numbers. In this model time is interpreted as a set of equally spaced time points, each time point t having a single successor. The present point of time, which advances as time goes by, is denoted by t_{now}. The extension of a class C at time point t is denoted by $Ext(C, t)$, the value of an object o for an attribute $A \in schema(C)$ at time point t is denoted by $Val_C(o, A, t)$. If no time point is explicitly specified, we assume the time point t_{now}. We

furthermore assume a function $Time_C(o, A, t)$ that determines the time point $t' \leq t$ the value $A.o$ of attribute A of object $o \in Ext(C, t)$ was updated the last time before t.

The above definitions and assumptions now provide us a suitable framework to define the data quality aspects *timeliness, completeness, and accuracy* in a formal way.

Definition 2.1 (Timeliness) Given two classes C_1 and C_2 with $schema(C_1) = schema(C_2)$. Class C_1 is said to be more *up-to-date* than C_2 at time point $t \leq t_{now}$ with respect to attribute $A \in schema(C_1)$, denoted by $C_1 >_{A,t}^{time} C_2$, iff

$\text{count}\{o_1 \mid o_1\colon Ext(C_1, t), o_2\colon Ext(C_2, t) : same(o_1, o_2) \wedge time_C(o_1, A, t) > time_C(o_2, A, t)\} \geq$
$\text{count}\{o_2 \mid o_1\colon Ext(C_1, t), o_2\colon Ext(C_2, t) : same(o_1, o_2) \wedge time_C(o_2, A, t) > time_C(o_1, A, t)\}$

■

In other words, class C_1 is more up-to-date than C_2 at time point t with respect to attribute A if its extension $Ext(C_1, t)$ contains more recent updates on A than $Ext(C_2, t)$ does. Note that for $t = t_{now}$ this property may flip since t_{now} advances as time goes by. It should also be noted that there are alternative definitions for timeliness. Possible definitions depend on how much information about update strategies is available for local databases and classes at integration time. For example, focusing more on the distances between the time points where the attribute values of the same objects were updated, would give rise to the following condition:

$sum\{time_C(o_1, A, t) - time_C(o_2, A, t) | o_1\colon Ext(C_1, t), o_2\colon Ext(C_2, t) : same(o_1, o_2)\} \geq$
$sum\{time_C(o_2, A, t) - time_C(o_1, A, t) | o_1\colon Ext(C_1, t), o_2\colon Ext(C_2, t) : same(o_1, o_2)\}$

The following example shows the difference between the two conditions.

Example 2.2 Assume that the following values for $time_{C_1}$ and $time_{C_2}$ have been determined at $t = t_{now} = $ '10-14-98' for the attribute A. The attribute values $A.C_1$ and $A.C_2$ shown in each row are assumed to be values of objects from $Ext(C_1, t)$ and $Ext(C_2, t)$ and both objects refer to the same object in C_{virt}.

$A.C_1$	$time_{C_1}(o_1, A, t)$		$A.C_2$	$time_{C_2}(o_2, A, t)$
120	10-9-98		125	10-3-98
156	10-9-98		156	10-3-98
123	10-7-98		130	10-8-98
108	10-7-98		111	10-8-98

Applying the first condition would yield that $C_1 =_{A,t}^{time} C_2$ holds, while for the second condition we have $C_1 \geq_{A,t}^{time} C_2$ ($12 > 2$). ■

Although at a local database modifications of real world objects are mapped in a timely manner using a certain data capturing approach, this does not necessarily mean that this approach suitably propagates information about deleted or new objects of the virtual class. While timeliness essentially refers to properties of attributes, the data quality aspect *completeness* focuses on the extensions of the two classes C_1 and C_2 as a whole.

Definition 2.3 (Completeness) Class C_1 is said to be *more complete* than the class C_2 at time point $t \leq t_{now}$, denoted by $C_1 >_t^{comp} C_2$, iff

$\text{count}\{o_1 | o_1 \in Ext(C_1, t) \wedge \neg \exists o' \in Ext(C_{virt}, t) : same(o_1, o')\} <$
$\text{count}\{o_2 | o_2 \in Ext(C_2, t) \wedge \neg \exists o' \in Ext(C_{virt}, t) : same(o_2, o')\}$

■

In other words, although the extension of C_2 may contain more objects at time point t, this does not necessarily mean that these objects still exist in the corresponding virtual class. C_2 can contain numerous objects which already have been deleted in reality at a time point $t' < t$. The above definition thus nicely reflects the aspect of outdated objects. Note that we have not included the aspect that C_1 must also contain more information about any real world object from C_{virt} than C_2 does. Including this aspect would imply that at a local database one is always interested in all objects that belong to a virtual class. This, however, is often not the case because the *working scope* of local classes and applications is typically restricted to a subset of such real world objects.

The third data quality aspect, which is orthogonal to timeliness and completeness, is the aspect of *data accuracy* which focuses on *how well* properties or attributes of real world objects are mapped into local classes.

Definition 2.4 (Data Accuracy) Given two classes C_1 and C_2 with $schema(C_1) = schema(C_2)$ and attribute $A \in schema(C_1)$. Class C_1 is said to be *more accurate* than C_2 with respect to A at time point t, denoted by $C_1 >_{A,t}^{acc} C_2$, iff

$$\text{count}\{o_1 \mid o_1 \in Ext(C_1, t), o_2 \in Ext(C_2, t), o \in Ext(C_{virt}, t) : same(o_1, o_2) \wedge same(o_1, o) \wedge$$
$$|Val_{C_{virt}}(o, A, t) \stackrel{\circ}{-} Val_{C_1}(o_1, A, t)| \leq |Val_{C_{virt}}(o, A, t) \stackrel{\circ}{-} Val_{C_2}(o_2, A, t)| \} >$$
$$\text{count}\{o_2 \mid o_1 \in Ext(C_1, t), o_2 \in Ext(C_2, t), o \in Ext(C_{virt}, t) : same(o_1, o_2) \wedge same(o_2, o) \wedge$$
$$|Val_{C_{virt}}(o, A, t) \stackrel{\circ}{-} Val_{C_1}(o_2, A, t)| \leq |Val_{C_{virt}}(o, A, t) \stackrel{\circ}{-}_{C_2}(o_1, A, t)| \} \qquad \blacksquare$$

In the above definition $\stackrel{\circ}{-}$ denotes a generic minus operator which is applicable to either a pair of strings, numbers or dates. In order to suitably incorporate the aspect of possible *null* values, one can define a value a_{max} that is used if $Val_{C_i}(o_i, A, t)$ is *null*. It is even possible to give a definition for data accuracy that takes only the number of objects into account that have the value *null* for the attribute A.

Determining the timeliness and accuracy of data makes only sense for certain attributes. For example, object identifiers are typically time-invariant and thus cannot cause any data quality problems (unless a property of real world objects designates the object identifier). We will discuss the aspect of time-varying and time-invariant attributes in more detail in Section 3. The important point with the above definitions is that they describe orthogonal data quality aspects. That is, in case of a data conflict among two objects referring to the same real world object, it is possible to choose either the most accurate or the most up-to-date data about this object, depending on whether respective specifications exist for the two objects. Also in case of non-conflicting object classes and extensions (i.e., the extensions of semantically equivalent local classes are disjoint), a data quality aspect might give a reason to choose one extension over the other or, at least, to distinguish or group the results retrieved from these classes. Both scenarios, of course, require to weaken the transparency property of integrated classes and objects at the global level.

In practice it is rather unlikely that there is one class among several semantically equivalent classes such that this class satisfies all three data quality aspects best. That is why these, possibly time-varying, aspects need to be modeled suitably and utilized for global query processing.

3 Modeling Data Quality Aspects

In the previous section we have shown that it is possible to formally define data quality aspects using virtual classes. In practice, however, virtual classes are not explicitly provided in a way that they can easily be used to evaluate the conditions given in Definitions 2.1, 2.3, and 2.4. In this section we discuss how statements about data quality can be extracted from local database participating in a multidatabase system. We show how these statements can be modeled as metadata available for global query processing. The underlying concept for this is *information profiling*.

3.1 Information Profiling

Information profiling essentially refers to the activity of describing the information capturing and processing techniques used to populate real world data into a local data model and data structures, respectively, classes. Information profiling is not peculiar to database integration, but is an integral task in database and application design. An information profile, e.g., for a class, describes not only the schema of that class (static properties) but also how real world data or artifacts are mapped as objects into this class (dynamic properties). This includes a description of (sets of) object methods and how these methods interact with the database and application environment. Most information for profiling can be obtained while structural and semantic conflicts are investigated at multidatabase design time. As any other proposal to database integration and resolving semantic conflicts, resolving operational heterogeneity through information profiling requires not only a good knowledge about particular local database schemas but also about the environments in which the local databases operate.

Given a local class C with attributes $\langle A_1, \ldots, A_n \rangle$ and a data quality aspect $Q \in$ {timeliness, completeness, accuracy}. The basic idea for information profiling is to partition C and its possible extensions $PExt(C)$ into a set $\mathcal{U}_C$ of *information units* such that all (possible) data in a unit $u \in \mathcal{U}_C$ show the same properties (see below) with respect to Q. A *basic information unit* u for a class C and its possible extensions $PExt(C)$ can be one of the following types. We assume that the object identifier is part of every unit.

(U1) an attribute $A \in \{A_i, \ldots, A_k\}$, i.e., $u^A = A$

(U2) a subset of possible attribute values for an attribute $A \in \{A_i, \ldots, A_k\}$,
i.e., $u^A = \{\pi_A(o) \mid o \in PExt(C) \wedge p(o)\}$ with p being a selection predicate.

The set of basic information units $u_1^A, u_2^A, \ldots, u_k^A$ associated with an attribute $A \in schema(C)$ is denoted by U_C^A. Partitioning a class C into basic information units can be optimized by building *composed information units* that consider more than one attribute, i.e., $u = \{A_{i1}, \ldots, A_{ik}\}$ or $u = \{\pi_{A_{i1}, \ldots, A_{ik}}(o) \mid o \in PExt(C) \wedge p(o)\}$, where $A_{ij} \in \{A_i, \ldots, A_k\}$, if all the constituting basic units have the same properties with respect to the data quality aspect Q. In order to discuss the basic idea of information profiling and also later the global query processing in the presence of information units (Section 4), we do not consider these optimization issues in the rest of the paper.

With an information unit a profile is associated, describing how and when the data in this unit are captured, processed and maintained. Such descriptions, of course, depend on

9

the type of data quality aspects Q for which the class C has been partitioned (*Q-profile*, *Q-partitioning*). From a practical point of view, we think that it is rather unlikely to have a formal specification for profiles. Depending on the type of data quality aspect, a profile should consider, respectively address, the following issues:

- *Timeliness* Given an attribute $A \in schema(C)$. What is the (average) update frequency of that attribute? Do updates occur on an event-based basis? Are update frequencies time-dependent?

- *Completeness* How do data processing techniques ensure that real world objects relevant for that class are recorded? Are the employed processing techniques time-dependent?

- *Data Accuracy* What type of technique is used to record real world data? Are data entered manually or are they read, e.g., by sensors? Are the data extracted from other resources? How many data processing units (applications) are involved in reading real world data into local data structures? Are the employed techniques time-dependent?

We are fully aware of the fact that not having a formal specification for data quality profiles can be a critical point. It should, however, be noted that formal specifications often do not exist where structural or certain semantic relationships and heterogeneities among classes and attributes need to be discovered and resolved at the database schema level. We think that system descriptions such as data flow diagrams, workflow diagrams, or even information obtained through system analysis can often provide useful and sufficient data for information profiling.

Beside the information units U_C^A that can be associated with an attribute $A \in schema(C)$ there are two special types of units. The first one we call *default unit*, denoted by u_d^A, and it corresponds to the partition of A for which data is (or will be) recorded but no Q-profile can be given. The second type, denoted by u_0^A, is the description of the partition for which data in $PExt(C)$ exist, but for which data are never recorded because, e.g., they are not admissible due to some integrity constraints. Note that (1) the specification of u_A^0 can always be determined based on the specifications of the other units, and (2) the specification of u_0^A is always of type (U2). Adding these two types of units to U_C^A for which profiles have been determined leads to the fact that for an attribute A and associated domain D always a complete partitioning can be given, which, in the worst case (no profiles available), consists only of the default unit u_d^A.

3.2   Class Partitioning and Quality Assertions

Partitioning and information profiling is applied to all attributes of a class C. That is, for C a set $U_C^A = U_C^{A_1} \cup \ldots \cup U_C^{A_n}$ of information units is determined. Applying this technique to some sample classes and their possible extensions shows that time-invariant attributes typically are all partitioned in the same way (i.e., same specification of p for basic units of type (U2)) while partitions for time-variant attributes (i.e., attributes that are updated frequently) differ. Figure 2 illustrates an example result of partitioning a class C with time-invariant attributes A_1, A_3 and time-variant attributes A_2, A_4.
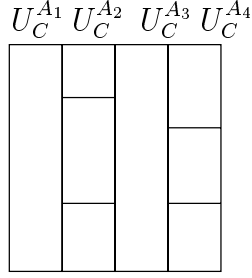
$$U_C^{A_1} \; U_C^{A_2} \;\; U_C^{A_3} \; U_C^{A_4}$$

Figure 2: Partitioning of a class C

Having at least two partitions u_i^A and u_j^A different from u_0^A and u_d^A for an attribute A indicates that even within possible extensions of C, the quality of the data values for A differs. This is quite natural since we cannot expect all objects and attributes in a class to have the same quality with respect to a data quality aspect Q. Because the partitions itself do not say much about how the quality of the data in these units is related, *attribute quality assertions* are determined among the units in U_C^A and associated profiles such that a complete order among the units in U_C^A is obtained. Determining this order is done by pairwise comparing the Q-profiles associated with each unit $u_i^A \in U_C^A \backslash \{u_0^A, u_d^A\}$.

Definition 3.1 (Attribute Quality Assertions) Given two information units $u_i^A, u_j^A \in U_C^A \backslash \{u_0^A, u_d^A\}$ which have been determined based on a Q-partitioning of A. If the profiles associated with u_i^A, u_j^A reveal that the data contained in u_i^A has a better quality than the data contained in u_j^A, then the attribute quality assertion $u_i^A >^Q u_j^A$ is said to be valid. ∎

The goal of pairwise comparing profiles associated with the u_i^As is to determine a complete order among the units in U_C^A. Not having a complete order but only a partial order introduces some kind of vagueness in the assertions which then has to be dealt with in special way. For the sake of simplicity and also due to space limitations, in the remainder of the paper we assume that always a complete order can be determined. Note that u_d^A cannot be compared with any other unit from U_C^A because no profile for u_d^A is given, thus there are no assertions referring to u_d^A. Furthermore, there is no need to specify any assertion involving u_0^A because corresponding data is never recorded for that unit in C.

Before we discuss comparisons of attribute quality assertions of semantically equivalent classes, we first focus on the aspect of time which we explicitly introduced in Section 2.2. Time plays an important role when determining quality assertions among units that have been obtained based on a *timeliness*-partitioning (see also scenario 1 in Section 1.1). It is easy to see that if time plays a role, the total order among information units in U_C^A is time-dependent, too. While a time-dependent order can be optional with respect to the data quality aspects completeness and accuracy, it is intrinsic to the aspect of timeliness. This means that for each possible time point t, we have to specify a total order among the units in U_C^A. This can be done in different ways. Assuming a finite domain for time, e.g., number of the week in a year or day of the week, it is possible to give a complete specification of time-dependent orders. Thus with each order a set of days or numbers of weeks can be associated. More complex temporal quantifiers for time-dependent orders, of course, can be useful and need to be investigated in future research. Here we assume that with each order a set of time intervals or time points is associated and that, given a

partitioning U_C^A and order among the informations units in U_C^A , with each time point t an order can be associated.

3.3 Inter-Class Considerations

Assume that two semantically equivalent classes $C_1\langle A_1, \ldots, A_n\rangle$ and $C_2\langle A_1, \ldots, A_n\rangle$ have been Q-partitioned and that all total orders have been determined for the units in $U_{C_1}^{A_1}, \ldots, U_{C_1}^{A_n}$ and $U_{C_2}^{A_1}, \ldots, U_{C_2}^{A_n}$. The task now is to *combine* the partition sets $U_{C_1}^{A_i}$ and $U_{C_2}^{A_i}$ into a new partitioning $U_C^{A_i}$ such that C is the global class integrating C_1 and C_2. In building U_C^A the task is not to define the structure of C but the data integration rules associated with C at the global level. The main idea for this is to resolve possible data conflicts among semantically equivalent objects in C_1 and C_2 by comparing and ordering the information units and profiles that may contain (attributes of) conflicting objects. Also in case no data conflicts will exist among possible extensions of C_1 and C_2, the information units and profiles are used to establish an order (with respect to a data quality aspect Q) among possible objects in the global class C.

Given two information units $U_{C_1}^{A_i}$ and $U_{C_2}^{A_i}$, a portioning $U_C^{A_i}$ is obtained by *overlapping* the units in $U_{C_1}^{A_i}$ with the units $U_{C_2}^{A_i}$ as illustrated in Figure 3.
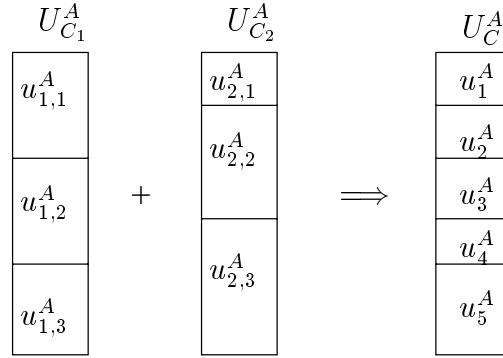

Figure 3: Overlapping of local information units

Note that the specification of the units in U_C^A can easily be obtained from the specification of the units in $U_{C_1}^A$ and $U_{C_2}^A$. Furthermore, the partitioning U_C^A is always complete because of the default units in $U_{C_1}^A$ and $U_{C_2}^A$. Given the specifications of the partitions in U_C^A, the next step is to compare the profiles associated with the (local) units that contribute to a unit $u_i^A \in U_C^A$. In Figure 3, for example, the units that contribute to $u_1^A \in U_C^A$ are $u_{1,1}^A \in U_{C_1}^A$ and $u_{2,1}^A \in U_{C_2}^A$, and for u_2^A they are $u_{1,1}^A$ and $u_{2,2}^A$. Because the profiles associated with the local units all refer to the same data quality aspect Q, it is possible to determine a preference among two local units that contribute to a global unit in U_C^A. Determining preferences essentially corresponds to resolving data conflicts in case for a global object there is a corresponding object in each local unit. Again, the two types of local information units $u_{i,d}^A$ and $u_{i,O}^A, i \in \{1, 2\}$ need special consideration. We consider the different cases assuming that $u_{i,d}^A = u_{1,d}^A$ and $u_{i,0}^A = u_{1,0}^A$ and that either of these two units is compared with a unit $u_{2,j}^A \in U_{C_2}^A$:

1. $u_{1,d}^A$ and $u_{2,j}^A \in U_{C_2}^A \setminus \{u_{2,d}^A, u_{2,0}^A\}$: because no profile is associated with $u_{1,d}^A$, no preference can be given unless it is known that, for example, $u_{2,j}^A$ has the highest quality

among the local units in $U_{C_2}^A$ with respect to the data quality aspect Q.

2. $u_{1,d}^A$ and $u_{2,d}^A$: no preference can be given, meaning that it is possible at the global level to return two local objects (or attributes) that correspond to the same global object and there (perhaps) is a data conflict among the two objects (attributes) in $u_{1,d}^A$ and $u_{2,d}^A$.

3. $u_{1,0}^A$ and $u_{2,j}^A \notin \{u_{2,d}^A, u_{2,0}^A\}$: Because $u_{1,0}^A$ never contains data, $u_{2,j}^A$ is chosen.

4. $u_{1,0}^A$ and $u_{2,d}^A$: Same as for 3.

5. $u_{1,0}^A$ and $u_{2,0}^A$: There are never data in either of the two units. Thus there will never be a conflict that needs to be resolved based on the profiles associated with the two units.

Once a preference among the units building the global information units in U_C^A has been determined, again a complete order among the information units in $U_C^A \backslash \{u_d^A, u_0^A\}$ is determined. Note that in case the two local classes C_1 and C_2 have been partitioned with respect to a data quality aspect Q and the total order among the units in $U_{C_1}^A$, respectively $U_{C_2}^A$, is time-dependent, the total order among the units in U_C^A can be time-dependent, too. More importantly in this case, comparisons of the profiles of local units must occur for each time point t. It remains to be investigated to what extent the process of determining a (time-dependent) total order among the units in U_C^A can be determined automatically, given the attribute quality assertions for $U_{C_1}^A$ and $U_{C_2}^A$.

Before we conclude this section by describing how information units and profiles are recorded as metadata for global query processing, we first take a closer look at the data quality aspect completeness. As described in Section 2.2, completeness refers to complete extensions of classes rather than on possible data conflicts covered by the aspects timeliness and accuracy. Partitioning a class and its possible extensions with respect to the aspect completeness often reveals a very simple pattern. Figure 4 illustrates an optimized completeness-partitioning $U_{C_1}^A$ and $U_{C_2}^A$ for two semantically equivalent local classes C_1 and C_2.
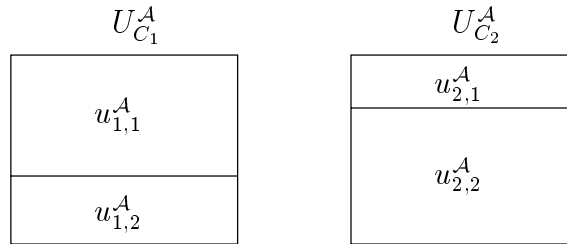

Figure 4: Partitioning based on the data quality aspect *completeness*

Note that for all attributes in $\mathcal{A}$ the same partitioning is adopted. This is quite natural because completeness refers to objects as a whole and not to single attributes. For this particular data quality aspect thus defining local attribute quality assertions as well as defining the information obtained through overlapping of local units can be done easily.

3.4 Data Quality Information as Metadata

As mentioned earlier, information about partitioning and overlapping semantically equivalent classes is used by the global query processor to deal with resolving data conflicts and ordering objects and attributes of different quality.

The metadata repository utilized by the query processor already contains information about the structure of global (conceptual) classes and the (structural) mappings of local classes and attributes into global classes. Beside this structural information about metadata at the global level, the repository also has to record information about how to actually integrate data contained in local class extensions into global class extensions. Such data integration rules are typically specified by using local class structures formulated in the global data model. Assume two semantically equivalent classes C_1 and C_2 where structural and (certain) semantic conflicts have been resolved, and C_1, C_2 are specified in the global data model. A simple data integration rule for the global class C then might look like $Ext(C) := Ext(C_1) \cup Ext(C_2)$ (using the relational operator union). Depending on possible data conflicts among objects in $Ext(C_1)$ and $Ext(C_2)$, a data integration rule can be more complex because conflict resolution functions are encoded in the rule.

How do data integration rules look like in the presence of various information about data quality of local classes, information units etc? While for traditional approaches to query processing in multidatabase systems the global query processing engine simply utilizes the specified data integration rule, for the scenario presented in this paper it is the task of the query engine to form a data integration rule. For this, the following information is recorded in the metadata repository at the integration layer.

- For each local class C_i and each data quality aspect Q, the specification of the Q-partitions, the associated Q-profiles, the description of the attribute quality assertions, and their (time-dependent) total order.

- For each global class C and each attribute $A \in schema(C)$, the specification of the partitioning U_C^A and the description of the total order among the units in U_C^A.

A more precise definition of the information recorded in the metadata repository depends on the underlying global data model, the type of expressions allowed in building information units of type (U2), and the type of temporal quantifiers that can be associated with total orders among information units. For the sake of simplicity, in the following section we assume the abstract description of the metadata as above.

4 Query Processing

In this section we give a short outline of the aspect of global query processing in a multidatabase system in the presence of data quality information. Because of the various features and possible optimization techniques that can be applied in this scenario, we only give the basic idea of global query processing. For example, we do not consider complex queries (nested queries, subqueries). We also assume that the reader is familiar with the basic tasks and methods in query processing in multidatabase systems (see, e.g., [MY95] or [EDN97]).

4.1 Global Query Language

A main feature of a global query language for a MDBS should be to provide global users, designers and applications a transparent access to local data. That is, the user should be unaware of possible structural and semantic conflicts among local meta-data, and s/he should also not be responsible for resolving such conflicts (some approaches, e.g., [SRL93, MR95] adopt exactly the opposite strategy).

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 a prespecified unique data integration rules. But what does this mean when there are local data of different quality and the data need to be integrated at run-time? First, as discussed extensively, data quality aspects are orthogonal. That is, there does not exist a unified or unique view on the integrated data. Assuming that the user is aware of the fact that local data have different quality, query language constructs are necessary that support the specification of a *data quality goal* for global queries and thus integrated data. Such an approach then supports the following two aspects:

1. In case there are data conflicts among two or more semantically equivalent local objects, the object (or attributes) satisfying the specified data quality goal best is chosen. Note that in this case the metadata about local information units and preferences among these units (with respect to a global class) are utilized.

2. In case there are no conflicts among objects of local classes but the quality of the objects (with respect to a data quality aspect Q) is different, the integrated objects need to be grouped or tagged to indicate this aspect at the global query interface.

In order to support querying objects and attributes with possibly different quality, we suggest an extensions of a global query language, say OQL, by a data quality goal clause:

> **select** <list of attributes>
> **from** <list $C_1, \ldots, C_n$ of global classes>
> **where** <selection condition>
> **with goal** <data quality goal>;

A simple data quality goal can either be *most accurate*, *most up-to-date*, or *most complete*. It is also possible to allow a list of data quality goals, meaning that among two conflicting objects (attributes) which have the same quality with respect to the first goal, those attributes are chosen which satisfy the second goal best.

4.2 Querying a Global Class

Assume a global class $C\langle A_1, \ldots, A_4\rangle$ composed of local classes $C_1, \ldots, C_m$. In the most simple case, a query of the type

> **select** $C.*$ **from** C **with goal** most-accurate;

could retrieve local data following the data integration rule $Ext(C) = Ext(C_1) \cup \ldots \cup Ext(C_m)$ (C_i being specified in the global data model). Now assume that information profiling has been performed for the local classes with respect to the data quality aspect accuracy and the following pattern has been obtained for the global class C:
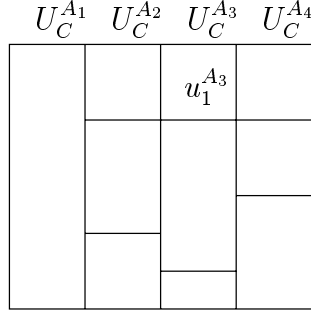
$$U_C^{A_1} \quad U_C^{A_2} \quad U_C^{A_3} \quad U_C^{A_4}$$

$$u_1^{A_3}$$

Figure 5: Global Class C with its information units

For the above query, the query processor then utilizes the specification of each information unit in $U_C^{\mathcal{A}}$ in order to compose global objects in $Ext(C)$ from local objects and attributes in $Ext(C_i)$. For example, for attribute values in the unit $u_1^{A_3}$, the query processor chooses those values from corresponding local units that satisfy the aspect accuracy best (according to the attribute quality assertions specified for the unit $u_1^{A_3}$). Roughly speaking, for each object identifier known to be in $Ext(C)$, corresponding attribute values are selected from possibly different local information units. Note that the object identifier is contained in each local information unit (see Section 3.1).

Note also that for another data quality aspect the partitioning $U_C^{\mathcal{A}}$ might look completely different. In particular, if the data quality goal timeliness has been specified in a global query, the point in time the global is issued determines which of the different time-dependent orders among the partitions in $U_C^{\mathcal{A}}$ is chosen.

In any case, if for an attribute A there exist multiple units within a global class and for these units attribute quality assertions have been specified, displaying attribute values suggests to indicate the variances among the quality of the attribute values. This, for example, can be done by using certain coloring schemas or object grouping structures. This feature is in particular necessary if a global user wants to compare the different results of the same queries with different data quality goals.

4.3   Joining Global Objects

In the presence of data quality information one would like to avoid simply combining objects that have different qualities (that is also why we suggest certain grouping structures in the previous section). For example, it should be avoided - or alternatively indicated in the query result - that an "up-to-date" object is joined with an outdated object (based on the relational join operator). Because joining object occurs in a similar way as joining tuples in relational databases, based on primary key and foreign key relationships, ensuring this property is quite trivial under the following assumption: Primary and foreign key attributes (or object references) always have the same quality for all data quality aspects. Obviously, this assumption can easily be verified during modeling data quality aspects as shown in Section 3. Suppose the query

> **select** $C_1.*$, $C_2.*$ **from** C_1, C_2
> **where** $C_1.primary\text{-}key = C_2.foreign\text{-}key$
> **with goal** most accurate;

where the join condition is based on a primary-foreign key relationship. For this query, the query processor first builds the extension of the global class C_2 according to the specified data quality goal. Then, for each referenced object, the respective (subset of the) extension of C_1 is built according to the same data quality goal, but restricted to the objects needed for $Ext(C_1)$.

5   Conclusions and Future Work

In this paper we have presented a novel framework to handling diverse data quality aspects in database integration. We have shown that there is a strong need for supporting data quality in data integration, ensuring that applications built on top of a multidatabase system can rely on "high-quality data".

The important contribution to database integration approaches in this paper is that we have shown (1) how to formalize the basic data quality aspects timeliness, completeness and accuracy, and (2) how to model diverse data quality properties of local and global class extensions using information units, information profiling, and attribute quality assertions. We are convinced that the used concept of information profiling plays an important role in designing a multidatabase system, in particular for designing data quality dependent data integration strategies and rules. In our future work we aim to develop specifications of information profiles which are more formal, ideally allowing to compare profiles and determining attribute quality assertions on an automated basis, supported by tools of, e.g., a multidatabase system design environment. Another aspect which needs more investigations is the use of a temporal logic framework for specifying time-dependent orders among attribute quality assertions.

Making the notion of data quality explicit at the multidatabase query language level opens up a completely new area of research in multidatabase query processing:

- How to exploit data quality features recorded for local and global classes and their extensions at the global query level? What are useful query language features?

- How to perform multidatabase query optimizations in the presence of data having different quality?

- How to represent data of different quality at the global query interface level?

Having a global query processing engine that utilizes metadata about information profiles, attribute quality assertions and preferences among information units provides a very flexible means to cope with dynamic database environments. That is, if profiles for local databases and information units change, these changes need only to be described at the metadata level. More precisely, no new data integration rules need to be specified at that level but they are dynamically determined by the query processing engine.

Further weakening the transparency of the existence of local databases at the global level can provide application designers and users a sophisticated means to perform data cleansing. Because differences among the quality of local data is suitably represented at the global level, query language constructs might be useful to investigate the source of poor quality data. Thus, the framework presented in this paper furthermore provides a suitable basis for applying data cleansing techniques to local database. The development of such environments and underlying strategies in the context of multidatabase systems and in particular data warehouses are subject to our future research.

References

[BA97]   J. Bischoff, T. Alexander: *Data Warehouses: Practical Advice from the Experts.*
            Prentice-Hall, 1997.

[BE96]   O. Bukhres, A. Elmagarmid: *Object Oriented Multidatabase Systems.* Prentice-Hall,
            1996.

[BLN86]   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.

[CTK96]   A. L. P. Chan, P. Tsai, J.-L. Koh: Identifying Object Isomerism in Multidatabase
            Systems. *Distributed and Parallel Database Systems 4:2 (April 1996),* 143–168.

[DeM89]   L.G. DeMichiel: Resolving Database Incompatibility: An Approach to Performing
            Relational Operations over Mismatched Domains. *IEEE Transactions on Knowledge
            and Data Engineering, 1(4), December 1998* 485–493.

[EDN97]   C. Evrendilek, A. Dogac, S. Nural, F. Ozcan: Multidatabase Query Optimization.
            Distributed and Parallel Databases 5 (1997), 77–114.

[GMS94]   C.H. Goh, S.E. Madnick, M. Siegel: Context Interchange: Overcoming the Chal-
            lenges of Large-Scale Interoperable Database Systems in a Dynamic Environment.
            In *Proceedings of the Third International Conference on Information and Knowledge
            Management (CIKM'94),* ACM Press, 1994, 337-346.

[GSC96]   M. Garcia-Solaco, F. Saltor, M. Castellanos: Semantic Heterogeneity in Multi-
            databases. Invited chapter in: O. Bukhres and A. Elmagarmid (eds.) *Object Oriented
            Multidatabase Systems.* Prentice-Hall, 129–202, 1996.

[Huf96]   D. Hufford: Data Warehouse Quality, *Data Management Review, Feb/Mar 1996*

[JJQ98]   M. Jarke, M.A. Jeusfeld, C. Quix, P. Vassiliadis: Architecture and Quality in Data
            Warehouses. 93-113 In *Advanced Information Systems Engineering – CAiSE'98.*
            LNCS Vol. 1413, Springer, 1998, 93-113.

[JV97]   M. Jarke, Y. Vassiliou: Data Warehouse Quality Design: A Review of the DWQ
            Project. Invited Paper, *Proc. of the 2nd Conference on Information Quality.* Mas-
            sachusetts Institute of Technology, Cambridge, 1997.

[Kim95]   W. Kim: *Modern Database Systems: The Object Model, Interoperability, and Be-
            yond,* 649–663. ACM Press, New York, 1995.

[Kim96]   R. Kimball: *The Data Warehouse Toolkit,* John Wiley, 1996.

[Kim96b]   R. Kimball: Dealing with Dirty Data. *DBMS Magazine 9:10, September 1996,* Miller
            Freeman, Inc., 1996.

[KCG95]   W. Kim, I. Choi, S. Gala, M. Scheevel: On Resolving Schematic Heterogeneity in
            Multidatabase Systems. In [Kim95], 521–550.

[KS91]   W. Kim, J. Seo: Classifying Schematic and Data Heterogeneity in Multidatabase
            Systems. *IEEE Computer 24:12 (December 1991),* 12–18.

[KS96]   V. Kashyap, A. Sheth, Semantic and Schematic Similarities Between Database Ob-
            jects: A Context-Based Approach. *The VLDB Journal, 5(4), Dec 1996,* 276–304.

[MR95]   P. Missier, M. Rusinkiewicz: 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.

[MY95]   W. Meng, C. Yu: Query Processing in Heterogeneous Environment. In [Kim95], 551–572.

[Pu91]   C. Pu: Key Equivalence in Heterogeneous Databases. In Y. Kambayashi and M. Rusinkiewicz and A. Sheth (eds.), *Proc. of the 1st Int. Workshop on Interoperability in Multidatabase Systems (IMS'91), Kyoto, Japan*, IEEE Computer Society Press, 314–316, 1991.

[Red96]   T.C. Redman: *Data Quality for the Information Age.* Artech House, Boston, 1996.

[RW95]   M. P. Reddy, R. Y. Wang: Estimating Data Accuracy in a Federated Database Environment. In S. Bhalla (ed.), *Information Systems and Data Management, Proc. of the 6th Conf., CISMOD'95*, LNCS 1006, Springer-Verlag, 115–134, 1995.

[She91]   A. Sheth: Semantic Issues in Multidatabase Systems, SIGMOD Record 20(4), Special Issue, December 1992.

[SG93]   F. Saltor, M. Garcia-Solaco: Diversity with Cooperation in Database Schemata: Semantic Relativism. *Proceedings of the 14th International Conference on Information Systems (ICIS'93, Orlando 1993)*, 247-254.

[SK93]   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.

[SL90]   A. Sheth, J. Larson: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys 22:3 (1990)*, 183–236.

[SRL93]   L. Suardi, M. Rusinkiewicz, W. Litwin: Execution of Extended Multidatabase SQL. In A. Elmagarmid, E. Neuhold (eds.), *Proc. of the 9th International Conference on Data Engineering - 1993*, 641–650, IEEE Computer Society Press, 1993.

[SPD92]   S. Spaccapietra, C. Parent, Y. Dupont: Model Independent Assertions for Integration of Heterogeneous Schemas. *VLDB Journal 1:1 1994)*, 81–126, 1992.

[SS95]   I. Schmitt, G. Saake: Managing Object Identity in Federated Database Systems. In M. Papazoglou (ed.), OOER'95: Object-Oriented and Entity-Relationship Modeling, Proc. of the 14th Int. Conf., Gold Coast, Australia, LNCS 1021, Springer-Verlag, Pages 400-411, December 1995.

[Wan98]   R.Y. Wang: A Product Perspective on Total Data Quality Management. *Communications of the ACM 41:2*, 58–65, 1998.

[WS96]   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.

[WSF96]   R.Y. Wang, V.C. Storey, C.P. Firth: A Framework for Analysis of Data Quality Research. *IEEE Transactions on Knowledge and Data Engineering 7:4 (August 1995)*, 623–640, 1996.