ELSEVIER

Data & Knowledge Engineering 27 (1998) 231–248

DATA & KNOWLEDGE ENGINEERING

Integration of relations with conflicting schema structures in heterogeneous database systems[†]

Frank S.C. Tseng[a,*], Jeng-Jye Chiang[b], Wei-Pang Yang[b]

[a]*Dept. of Information Management, National Institute of Technology at Kaohsiung, YenChao, Kaohsiung, Taiwan, 824, ROC*

[b]*Dept. of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, 30050, ROC*

Received 14 August 1995; revised 10 January 1997; accepted 3 October 1997

Abstract

Schema integration is an important discipline for constructing a heterogeneous database system, since there exists various semantic discrepancies among local schemas. However, prior works rarely address the problems of integrating local relations with conflicts of different schema structures. In this paper, we propose an approach for the integration of local relations with such conflicts. In our approach, we define some auxiliary data called *metadata* for local schemas. The relations can be classified into some classes according to the metadata. The local relations with conflicts of different schema structures are respectively transformed to new relations with the same structure. Finally, we integrate these new relations. We also explore the query decomposition process for global queries. A global query is decomposed into many local queries according to the local schemas' structures. The results of the local queries will be sent to the global site. Finally, the global site integrates the local results into a new one and poses the global query on the integrated result again to get the final result. Since this process filters out unnecessary data in each local site, it can also be regarded as an important work for global query optimization.

Keywords: Heterogeneous database systems; Schema integration; Metadata; Query decomposition; Global query optimization

1. Introduction

The advance in communication and database technologies has changed the data processing capabilities tremendously. The proliferation of independent databases in a distributed environment implies that, for effective information sharing, an increasing number of applications are required to

* Corresponding author. E-mail: imfrank @ccms.nitk.edu.tw

[†] This research was supported by the National Science Council, Taiwan, ROC, under contract no. NSC 86-2213-E-155-034.

PII: S0169-023X(98)00005-6

access and derive data from multiple independent databases. However, for independent databases, the data sources are created and developed independently; that is, they are pre-existing in an uncoordinated way without the consideration of future integration with other databases.

To derive data in such a *heterogeneous database environment*, prior works can be classified into two general approaches. One is to provide a global schema for the independent databases by integrating their schemas. Dayal and Hwang [10] and Motro [24] adopted this approach based on functional models, while Breitbart et al. [4] and Deen et al. [11] were based on relational models. Another variation of this approach does not require the creation of a global schema. On the other hand, for each application, the database administrator creates a schema describing only data that the application may access in the local databases. This type of system is also called a *federated database system* [14,15,16]. For a comprehensive survey of methodologies developed for schema integration, the gentle reader is referred to [1,3,6,17,19].

The other approach is by providing users a *multidatabase query language*. Users refer to the schemas and pose their queries against these schemas using the multidatabase query language. Litwin et al. [22,23] and Czejdo et al. [8] fell into this category. A multidatabase query language provides basic language constructs that allow users to issue queries across multiple database systems with heterogeneities. Grant et al. [13] has proposed a theoretical foundation for such languages by extending the relational algebra and calculus to a multi-relational algebra and calculus. In the following, *heterogeneous database systems* and *multidatabase systems* will be used interchangeably.

The schema integration process may present a large number of problems caused by various aspects of semantic discrepancy and design autonomy. Prior works of schema integration usually focus on resolving the incompatibility problems that may exist in different databases for semantically-related data [5].

In [18], schematic and data heterogeneity in heterogeneous database systems are systematically classified. Reddy et al. [25] also proposes a classification scheme for various kinds of semantic incompatibilities and data inconsistencies and presents a methodology that covers both schema integration and database integration. Moreover, Lee et al. [21] establishes a similar classification and proposes a way of optimizing multidatabase queries which takes advantage of the conflicts of schemas in searching for the execution plan with the least execution cost.

In this paper, we adopt the classification scheme proposed by Lee et al. [21], which categorizes the types of conflicts as follows.

(1) *Value-to-value conflicts:* These conflicts occur when databases use different representations for the same data. This type of conflict can be further distinguished into the following types:
   - *Data representation conflicts:* These conflicts occur when semantically related data items are represented in different data types.
   - *Data scaling conflicts:* These conflicts occur when semantically related data items are represented in different databases using different units of measure.
   - *Inconsistent data:* These conflicts occur when semantically related attributes for the same entity have different definite data values in different databases. Agarwal et al. [2] present a good work on addressing the problem of dealing with data inconsistencies while integrating data sets derived from multiple autonomous relational databases.

(2) *Value-to-attribute conflicts:* These conflicts occur when the same information is expressed as attribute values in one database and as attribute names in another database.

(3) *Value-to-table conflicts:* These conflicts occur when the attribute values in one database are expressed as table names in another database.
(4) *Attribute-to-attribute conflicts:* This occurs when semantically-related data items are named differently or semantically unrelated data items are named equivalently. The former case is also called *synonyms* and the latter case *homonyms* [25]. Some classification schemes call both cases *naming conflicts*.
(5) *Attribute-to-table conflicts:* These conflicts occur if an attribute name of a table in a database is represented as a table name in another database.
(6) *Table-to-table conflicts:* These conflicts occur when information of a set of semantically equivalent tables are represented in a different number of tables in another database. When integrating relations with such conflicts into a global relation, null values are usually generated. Such phenomenon is also called *missing data*.

These types of conflict can be further partitioned into the following two categories:
(1) Conflicts of similar schema structures: This category includes *value-to-value conflicts*, *attribute-to-attribute conflicts*, and *table-to-table conflicts*.
(2) *Conflicts of different schema structures:* This category includes *value-to-attribute conflicts*, *value-to-table conflicts*, and *attribute-to-table conflicts*.

For tables with conflicts of similar schema structures, an outerjoin operation [9] is usually employed to integrate the tables into a unified one. DeMichiel [12] has shown that some imprecise data, called *partial values*, may be derived due to scaling conflicts. We have also established some related result regarding the incompatibility problems and partial values. We developed some efficient algorithms to evaluate relational operations over partial values [7,28,29]. Besides, we generalized partial values into *probabilistic partial values* and proposed a general methodology to integrate relations with conflicts of similar schema structures [31]. Some properties that can be employed to refine partial values into more informative ones or even definite values were also being studied [30].

However, for the integration of tables with conflicts of different schema structures, most of the studies did not take into account the value-to-attribute conflicts, value-to-table conflicts, or attribute-to-table conflicts. In [20], Krishnamurthy et al. advocates some language features of a multidatabase query language should be added to cover these types of conflicts. In the following, we call such conflicts of different schema structures *structural conflict*—corresponding to tables with *value-to-attribute conflicts*, *value-to-table conflicts*, and *attribute-to-table conflicts*.

In this paper, we propose an approach to integrating tables with structural conflict. The problem can be informally stated as follows. The *domain* of an attribute of a table in Database A may be represented as a subset of the *attribute set* of another table in Database B, or in turn be a *set of table names* in Database C. To solve such structural conflict, Krishnamurthy et al. [20] has established the necessary language features for a multidatabase query language. Such approach conforms to the second aforementioned approach. In the following, we will resolve such problem through the other approach—the global schema approach. Our work proposes an integration process that integrates relations of structural conflict into a unified global schema. Besides, we also present the query decomposition rules for various situations.

In the rest of this paper, we will define three types of structural conflict in Section 2. Section 3 presents our integration process. Section 4 is devoted to the query decomposition rules. Finally, we conclude and present our future work in Section 5.

2. The problem of structural conflict

2.1. An example for informal illustrations

For a comprehensive presentation, we first give an example to demonstrate the problem of structural conflict. Then, we will define three types of structural conflict.

Example 2.1. Fig. 1 shows three databases containing the information for global stock markets. For the relation B in Site 2, suppose the attribute values of *Tokyo*, *HongKong*, *Taipei*, and *NewYork* in a record represent their index numbers of a date, respectively.

Although the three databases contain semantically equivalent data with respect to dates, cities and the index numbers, they appear to be in conflicting schema structures. The *domain* of *A.city* in Site 1 is represented as a subset of the *attribute set* of the relation B in Site 2. However, it is further represented as the set of *relation names* in Site 3. Such heterogeneity occurs when different databases are designed by different designers or by different considerations.

2.2. Metadata

Our schema integration process utilizes the semantic knowledge of all local schemas, which is called the *metadata* and should be prepared before integration. For a local relation, the metadata consist of the following components:

(1) *The domain of each attribute:* We use $Dom(A) = D$ to denote the domain of attribute A is D.

(2) *The semantic description of each attribute:* Ensuring autonomous systems to agree on the meaning of their exchanged data is not a trivial task. Semantic description often depends on

A

date	*city*	*index*	*currency*
10/5/92	Tokyo	24312	Yen
10/5/92	HongKong	3418	HK$
10/5/92	Taipei	3840	NT$
10/6/92	Tokyo	24420	Yen
10/6/92	HongKong	3431	HK$
10/6/92	Taipei	3852	NK$

Site 1

B

date	*Tokyo*	*HongKong*	*Taipei*	*NewYork*
10/5/92	24312	3418	3840	2579
10/6/92	24420	3431	3852	2584

Site 2

Tokyo

date	*index*	*quantity*
10/5/92	24312	568
10/6/92	24420	627

HongKong

date	*index*	*quantity*
10/5/92	3418	76
10/6/92	3431	64

Taipei

date	*index*	*quantity*
10/5/92	3840	135
10/6/92	3852	214

Site 3

Fig. 1. Three stock databases for Example 2.1.

context information, the database origin, the application, and so forth. We think how to formulate semantic information to facilitate interoperability among heterogeneous database systems is beyond the scope of our work. A very good formulation of such task has already been established by Sciore et al. [26]. They advocate *context information*, which is regarded as a metadata, must be an active component of information systems. The foundation of their work is based on the concept of a *semantic value*, which is defined to be a piece of data together with its associated context. To convert a semantic value from one context to another, *conversion functions* are employed.

To focus our discussion, we use a more simplistic notation adopted from [27]

$$Des(A) = \{S_1, S_2, \ldots, S_n\},$$

to denote the necessary descriptions (for schema integration) and to supply the semantics of attribute A, where S_1 is called the *primary description*. It is the 'actual' meaning of the attribute values. (Since an attribute name may be just an abbreviation or something not self-explanatory.) The others following the primary description are called the *auxiliary descriptions*. They are used to supply auxiliary information of attribute A. For example, if the 'actual' meaning of an attribute A is 'amount of money', then there may be an auxiliary description which represents 'unit of currency'. If there is no need of such primary description, then S_1 is replaced by an asterisk. For simplicity, we will use $Des^*(A)$ to denote the primary description of attribute A in the following (i.e. for the above example, $Des^*(A) = S_1$). Note that if the semantics of an attribute A is clear enough and needs no more descriptions, then $Des(A) = \emptyset$.

In Example 2.1, the domains of all involved attributes of each site are shown in Fig. 2, respectively. Besides, the semantic descriptions of each attribute are listed in Fig. 3, respectively. For example, in Fig. 3

$$Des(Tokyo) = \{Index, Yen\}$$

means '*Tokyo*' represents *Tokyo's index number* and its currency is *Yen*. Note that the semantics of the attribute *date* is clear enough. Therefore, its semantic description is $\emptyset$.

A primary description should be consistent with the naming (i.e. attribute names or relation names) of the other semantically related data located in the other sites. This is similar to resolving the *naming conflicts* (see Section 1) and should be determined manually.

Site 1: $Dom(date) = \{ 10/5/92 - 10/6/92 \}$,
$Dom(index) = \{ 0 - 100000 \}$,
$Dom(city) = \{ Tokyo, HongKong, Taipei, NewYork \}$,
$Dom(currency) = \{ US\$, Yen, HK\$, NT\$ \}$.

Site 2: $Dom(date) = \{ 10/5/92 - 10/6/92 \}$,
$Dom(Tokyo) = Dom(HongKong) = Dom(Taipei) = Dom(NewYork) = \{ 0 - 100000 \}$,
$Dom(quantity) = \{ 0 - 10000 \}$.

Site 3: $Dom(date) = \{ 10/5/92 - 10/6/92 \}$,
$Dom(index) = \{ 0 - 100000 \}$,
$Dom(quantity) = \{ 0 - 10000 \}$.

Fig. 2. The domains of all the attributes in Example 2.1.

Site 1: $Des(date) = Des(city) = Des(currency) = Des(index) = \emptyset$,
$Des(quantity) = \{ *, \text{million} \}$.
Site 2: $Des(date) = \emptyset$,
$Des(Tokyo) = \{ \text{index, Yen} \}$,
$Des(HongKong) = \{ \text{index, NH\$} \}$,
$Des(Taipei) = \{ \text{index, NT\$} \}$,
$Des(NewYork) = \{ \text{index, US\$} \}$.
Site 3: $Des(date) = Des(index) = \emptyset$,
$Des(quantity) = \{ *, \text{million} \}$.

Fig. 3. The semantic descriptions of all the attributes in Example 2.1.

In the following, we should ignore the resolutions for the conflicts of similar schema structures addressed in Section 1 and focus on resolving the structural conflict problem. We have proposed a general approach to resolving the conflicts of similar schema structures in [31].

2.3. Three types of structural conflict

In this section, three types of structural conflict are defined. We first define the following notations:
(1) $Sch(R)$ is used to represent the *attribute set* of a relation R.
(2) For two attributes A and B, we say A and B are *semantically equivalent*, denoted $A \leftrightarrow B$, if and only if

$$(A = B) \vee (A = Des^*(B)) \vee (B = Des^*(A)) \vee (Des^*(A) = Des^*(B)) .$$

Definition 2.1. For two distinct relations R and S, we define they belong to *Type-1 structural conflict* if and only if:

$$(\exists x \in Sch(R))((Dom(x) \cap Sch(S) \neq \emptyset) \wedge (\exists r \in Sch(R) - \{x\})(\exists s \in Sch(S) - [Dom(x) \cap Sch(S)])(r \leftrightarrow s) \wedge (\forall y \in Sch(S) \cap Dom(x))(\exists z \in Sch(R) - \{x\})(y \leftrightarrow z)) .$$

In Example 2.1, the relations ***A*** in Site 1 and ***B*** in Site 2 belong to Type − 1 structural conflict, since:
(1) the schema of ***A*** contains attribute *city*, such that $Dom(city) \cap Sch(B) \neq \emptyset$,
(2) both ***A*** and ***B*** contain the same attribute date, and
(3) for each element, say E, in

$$Sch(B) \cap Dom(city) = \{\text{Tokyo, HongKong, Taipei, NewYork}\}$$

there exists an attribute *index* in ***A***, such that E is semantically equivalent to *index*.

Definition 2.2. For two distinct relations R and S, we define they belong to *Type-2 structural conflict* if and only if:

$$\exists x \in Sch(R))(S \in Dom(x) \wedge (\exists r \in Sch(R) - \{x\})(\exists s \in Sch(S))(r \leftrightarrow s)) .$$

In Example 2.1, the relation ***A*** in Site 1 and any of the relations in Site 3 belong to Type-2 structural conflict, since:
(1) there exists an attribute *city* in A such that the set of relation names in Site 3 {Tokyo, HongKong, Taipei} $\subset Dom(city)$,

(2) there exists an attribute *date* in relation ***A*** and any of the relations in Site 3.

Definition 2.3. For two distinct relations R and S, we define they belong to *Type-3 structural conflict* if and only if:

$$(S \in Sch(R)) \wedge (\exists r \in Sch(R))(\exists s \in Sch(S))(r \leftrightarrow s)$$
$$\wedge (\exists y \in Sch(R))(\exists z \in Sch(S))(y = S \wedge y \leftrightarrow z).$$

In Example 2.1, the relation ***B*** and any of the relations in Site 3 belong to Type-3 structural conflict, since:

(1) all the relation names of Site 3 are in $Sch(\boldsymbol{B})$,
(2) there exists an attribute *date* in ***B*** and any of the relations in Site 3,
(3) for all the relations in Site 3, there exist an attribute *index* in their attribute sets, such that, for all the attributes *Tokyo*, *HongKong* and *Taipei* of relation ***B***, *index*↔*Tokyo*, *index*↔*HongKong* and *index*↔*Taipei*.

3. Transformation procedures for relations with conflicting schema structures

For the three types of structural conflict, we develop three procedures to transform relations with conflicting schema structures to a common one for integration.

3.1. Transformation procedure for type-1 structural conflict

For two relations R and S in Type-1 structural conflict, we know that an R's domain element is in the attribute set of S. Our procedure will transform relation S into a new one S', such that S' have the same structure with R. The formal procedure is as follows.

Procedure 1. *Type-1 transformation procedure:*
Input: Two relations, R and S, in Type-1 structural conflict.
Output: A new version of S, S', which has the same structure with R.

(1) For $x \in Sch(R)$, such that $Dom(x) \cap Sch(S) \neq \emptyset$, let $Dom(x) \cap Sch(S) = \{t_1, t_2, \ldots, t_k\}$.
(2) Split S into k relations, $S_1, S_2, \ldots, S_k$, where:
   - $S_i = t_i$,
   - $Sch(S_i) = Sch(S) - [Dom(x) \cap Sch(S)] \cup \{Des^*(t_i)\}$,
   - $Dom(Des^*(t_i)) = Dom(t_i)$, and
   - $Des^*(Des^*(t_i)) =$ '*', for $i = 1, 2, \ldots, k$

   That is, attribute t_i will be substituted by its primary description and the substituted attribute's primary description is changed to '*'.
(3) $Sch(S_i) \leftarrow Sch(S_i) \cup \{x\}$, where $Dom(x) = \{S_i\}$, for $i = 1, 2, \ldots, k$.
(4) $S' \leftarrow \cup_{i=1}^{k} S_i$.

Example 3.1. For the relations A and B in Example 2.1, we transform B into B' as Fig. 4 illustrates.

B

date	*Tokyo*	*HongKong*	*Taipei*	*NewYork*
10/5/92	24312	3418	3840	2579
10/6/92	24420	3431	3852	2584

(a) Original Relation B.

Tokyo

date	*index*
10/5/92	24312
10/6/92	24420

HongKong

date	*index*
10/5/92	3418
10/6/92	3431

Taipei

date	*index*
10/5/92	3840
10/6/92	3852

NewYork

date	*index*
10/5/92	2579
10/6/92	2584

(b) Split B into Four Subrelations.

Tokyo

date	*index*	*city*
10/5/92	24312	Tokyo
10/6/92	24420	Tokyo

HongKong

date	*index*	*city*
10/5/92	3418	HongKong
10/6/92	3431	HongKong

Taipei

date	*index*	*city*
10/5/92	3840	Taipei
10/6/92	3852	Taipei

NewYork

date	*index*	*city*
10/5/92	2579	NewYork
10/6/92	2584	NewYork

(c) Add to Each Subrelation an Attribute *city*.

B'

date	*index*	*city*
10/5/92	24312	Tokyo
10/5/92	3418	HongKong
10/5/92	3840	Taipei
10/5/92	2579	NewYork
10/6/92	24420	Tokyo
10/6/92	3431	HongKong
10/6/92	3852	Taipei
10/6/92	2584	NewYork

(d) Union the Four Subrelations.

Fig. 4. Type-1 transformation example.

3.2. Transformation procedure for Type-2 structural conflict

For two relations R and S in Type-2 structural conflict, we know that S's relation name is a domain element of one of R's attributes. Our procedure will transform S into a new one S', such that S' and R have the same structure. The formal procedure is as follows. If there are many S_i differ from R with structures in Type-2, our procedure will transform these relations at one time.

Procedure 2. *Type-2 transformation procedure:*
Input: Relations, R and S_i, $1 \leqslant i \leqslant n$, where R and each S_i are in Type-2 structural conflict.
Output: A single relation S', which is composed of all data of S_i, $1 \leqslant i \leqslant n$, and has the same structure with R.

(1) Let $x \in Sch(R)$, such that $S_i \in Dom(x)$, $i = 1, 2, \ldots, n$.
(2) $Sch(S_i) \leftarrow Sch(S_i) \cup \{x\}$, where $Dom(x) = \{S_i\}$, for $i = 1, 2, \ldots, n$.
(3) $S' \leftarrow \cup_{i=1}^{n} S_i$.

Example 3.2. For the relation A and those relations in Site 3 in Example 2.1, we transform the relations in Site 3 as Fig. 5 illustrates. Notice that although S' has a different attribute from that of A, A and S' can be integrated by an 'outerjoin' operation [9]. Please refer to [31] for more details.

3.3. Transformation procedure for Type-3 structural conflict

For two relations R and S in Type-3 structural conflict, we know that S's relation name is an attribute name of R. Our procedure will transform R into a set of relations R_j, such that each pair of R_j and S have the same structure. If there are many S_i different from R with structures in Type-3, our procedure will transform these relations at one time. The formal procedure is as follows.

Tokyo

date	*index*	*quantity*
10/5/92	24312	568
10/6/92	24420	627

HongKong

date	*index*	*quantity*
10/5/92	3418	76
10/6/92	3431	64

Taipei

date	*index*	*quantity*
10/5/92	3840	135
10/6/92	3852	214

(a) Original Relations in Site 3.

Tokyo

date	*index*	*quantity*	*city*
10/5/92	24312	568	Tokyo
10/6/92	24420	627	Tokyo

HongKong

date	*index*	*quantity*	*city*
10/5/92	3418	76	HongKong
10/6/92	3431	64	HongKong

Taipei

date	*index*	*quantity*	*city*
10/5/92	3840	135	Taipei
10/6/92	3852	214	Taipei

(b) Add an Attribute *city* into All the Relations in Site 3.

S'

date	*index*	*quantity*	*city*
10/5/92	24312	568	Tokyo
10/5/92	3418	76	HongKong
10/5/92	3840	135	Taipei
10/6/92	24420	627	Tokyo
10/6/92	3431	64	HongKong
10/6/92	3852	214	Taipei

(c) Union These Relations into S'.

Fig. 5. Type-2 transformation example.

Procedure 3. *Type-3 transformation procedure:*
Input: Relations R and S_i, $1 \leq i \leq n$, where each pair of R and S_i are in Type-3 structural conflict.
Output: A set of relations R_j, R_j and S_i have the same structure.

(1) Let $U = Sch(R) \cap \{S_1, S_2, \ldots, S_n\}$. Notice that U contains a subset of $Sch(R)$ and will be employed to vertically split R. However, U may be insufficient to split R into relations with the same structure of S_i. That is, there may be attributes in $Sch(R)$ that have the same primary description with those in $Sch(R) \cap \{S_1, S_2, \ldots, S_n\}$, but there is no corresponding relation name in S_i's site. These attributes should also be included in U to split R by checking if there are attributes in $Sch(R)$ that have the same primary description with those in $Sch(R) \cap \{S_1, S_2, \ldots, S_n\}$. Without loss of generality, we will assume U is sufficient to split R and denote $U = \{t_1, t_2, \ldots, t_k\}$ in this procedure.

(2) Split R into k relations $\{R_1, R_2, \ldots, R_k\}$, where
- $Sch(R_j) = (Sch(R) - U) \cup \{Des^*(t_j)\}$,
- $Dom(Des^*(t_j)) = Dom(t_j)$, and
- $Des^*(Des^*(t_j)) =$ '*', $1 \leq j \leq k$.

Example 3.3. For the relation B and those relations in Site 3 in Example 2.1, we transform the relation B as Fig. 6 illustrates.

3.4. Transformation for three types of structural conflict

In Example 2.1, the domain of attribute *city* of relation A (in Site 1) corresponds to a subset of relation B's attribute set (in Site 2) which in turn corresponds to a set of relation names in Site 3. In the following, we denote the relation set in Site 3 as a single relation C for simplicity. Besides, we regard A and B are in Type-1 structural conflict, A and C are in Type-2 structural conflict, and B and C are in Type-3 structural conflict.

B

date	*Tokyo*	*HongKong*	*Taipei*	*NewYork*
10/5/92	24312	3418	3840	2579
10/6/92	24420	3431	3852	2584

(a) The Original Relation B.

(b) $U = Sch(B) \cap \{Tokyo, Taipei, HongKong\} = \{Tokyo, Taipei, HongKong\}$.

(c) However, we found that *NewYork* should also be included in U.

Tokyo

date	*index*
10/5/92	24312
10/6/92	24420

HongKong

date	*index*
10/5/92	3418
10/6/92	3431

Taipei

date	*index*
10/5/92	3840
10/6/92	3852

NewYork

date	*index*
10/5/92	2579
10/6/92	2584

(d) Split B into Four Subrelations.

Fig. 6. Type-3 transformation example.

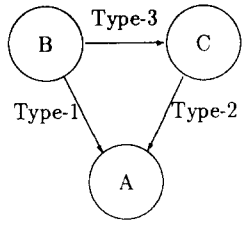


Fig. 7. A graphical illustration of three types of transformations.

By the above transformation procedures, we know that B can be transformed into A and C by Type-1 and Type-3 transformations, respectively. Besides, by Type-2 transformation, C can be transformed to A. This can be illustrated by Fig. 7.

If three types of structural conflict are present in a heterogeneous database system, our approach will choose A as the destination, then apply the above procedures to transform B and C into A, the common structure, and finally integrate them into a unified one.

Example 3.4. For example, in Example 2.1, since we have transformed **B** into **B′** (in Fig. 4d) and those relations in Site 3 into **S′** (in Fig. 5), we can integrate **A**, **B′** and **S′** into the global relation **A_B_C** depicted in Fig. 8.

Although it seems that B and C are also candidates to be the target of the transformation process, our approach will choose A (instead of B or C) as the destination. The reason is as follows. Suppose we choose B as the destination, then we will produce *null values* after the transformation when some tuples in A are absent (or some relations in C are absent). For illustration, if the relation A in Fig. 1 does not contain the following two tuples about the closing index in Hong Kong.

Date	City	Index	Currency
10/5/92	HongKong	3418	HK$
10/6/92	HongKong	3431	HK$

Then, after transforming A into A', a new version of A with the same structure of B, there will be null values in the attribute *HongKong*. As null values should be treated differently in database

A_B_C

date	*city*	*index*	*currency*	*quantity*
10/5/92	Tokyo	24312	Yen	568
10/5/92	HongKong	3418	HK$	76
10/5/92	Taipei	3840	NT$	135
10/6/92	Tokyo	24420	Yen	627
10/6/92	HongKong	3431	HK$	64
10/6/92	Taipei	3852	NK$	214

Fig. 8. The integrated relation **A_B_C** for Example 2.1.

management systems, which makes query processing more difficult and tedious, we recommend do not use B as the destination.

On the other hand, suppose we choose C as the destination, then we will decompose A (or B) into many relations like C. That makes our integration process less efficient, since the integration process is more efficient if it involves less relations. Therefore, we also recommend do not use C as the destination.

4. Query decomposition

After integrating local schemas into a global one, users can pose queries on the global schema. However, the data of the integrated schema should be derived from each local site. A brute force heterogeneous database system can acquire the whole relations from related local sites. However, such implementation suffers from its inefficiency. Therefore, we need to do global query optimization to speed up the processing speed.

That implies a query posed on a global schema should be decomposed into many local subqueries posed on their corresponding local schemas. Then, after receiving all answers from local sites, the global site integrates these answers and then pose the original global query to retrieve the desired data. This can be summarized by the following steps.

(1) The global site receives a global query from an end-user.
(2) It then decomposes the global query into a set of local queries and transmits each local query to its corresponding site.
(3) Each local site executes the received subquery and returns the result to the global site.
(4) After receiving all results of the subqueries, the global site integrates the results and poses the global query on the integrated result.
(5) Finally, the query result will be presented to the end-user.

In this section, we will consider the query decomposition for three types of structural conflict. Recall that for the situation depicted in Fig. 7, we will transform the structures of B and C into that of A for further integration. That is, A's structure is regarded as the global schema. Therefore, the query decomposition process will decompose a query posed on A's schema structure into queries posed on the schema structures of A, B and C.

To achieve this goal, we distinguish relation A's attributes into the following three categories:

(1) **Versatile attribute.** The attribute values of this type are so versatile such that they may appear in other sites as *attribute names* or *relation names*. For example, the attribute *city* of relation A in Example 2.1 belongs to this category.
(2) **Description attribute.** The attribute names of this type corresponds to the primary description of those attributes in $Dom(V) \cap Sch(B)$, where V is a versatile attribute in A. For example, the attribute *index* of relation A belongs to this category.
(3) **Non-versatile attribute.** An attribute that is not a versatile attribute nor description attribute. For example, the attribute *date* of relation A belongs to this category.

Based on these types of attribute, we will present the query decompositions for *selection*, *projection* and *join* operations in the following. Our discussion employs the situation depicted in Fig. 7.

4.1. Query decomposition for selection

To simplify our discussion, for a selection operation $\sigma_P(R)$, where R is an integrated relation and has the similar structure with A, we restrict the predicate P to involve a single attribute, say a, only. Therefore, a selection in the following can be specified as $\sigma_{P(a)}(R)$, $P(a) \equiv a\theta d$, $\theta \in \{>, <, \geqslant, \leqslant, \neq, =\}$. The decomposition depends on whether the involved attribute is a versatile, a description, or a non-versatile one.

(1) *The involved attribute is a versatile attribute.* For a global selection $\sigma_{P(a)}(R)$, $P \equiv a\theta d$, if a is a versatile attribute, then the subqueries for relations A, B and C are as follows:

- $A : \sigma_{P(a)}(A)$.
- $B : \begin{cases} \pi_{Com, S \cap Sch(B)}(B) & \text{if } \theta \in \{>, <, \geqslant, \leqslant, =\} \\ \pi_{Com, (Dom(a)-S) \cap Sch(B)}(B) & \text{if } \theta \in \{\neq\} \end{cases}$

  where:
  (a) *Com* is the set of common attributes of R and B and
  (b) S is the set of the domain elements of a that satisfy $P(a)$. That is $S = \pi_a(\sigma_{P(a)}(A))$ and can be obtained in the global site after receiving the result of the subquery posed on A.
- C : For each relation with relation name $C_i \in \pi_a[\sigma_{P(a)}(A)]$, pose the subquery $\sigma_{\text{true}}(C_i)$.

Example 4.1. Consider the global selection $\sigma_{\text{city}=\text{'Tokyo'}}(\mathbf{A\_B\_C})$ posed on the global relation **A_B_C** in Fig. 8. Then, the subqueries for A, B and C are:

- $A : \sigma_{\text{city}=\text{'Tokyo'}}(A)$,
- $B : \pi_{\text{date,Tokyo}}(B)$,
- $C : \sigma_{\text{true}}(\text{Tokyo})$.

(2) *The involved attribute is a description attribute.* For a global selection $\sigma_{P(a)}(R)$, if a is a description attribute, then the subqueries for A, B and C are:

- $A : \sigma_{P(a)}(A)$.
- B : If $Dom(a) \cap Sch(B) = \{x_1, x_2, \ldots, x_t\}$, then pose $\pi_{Com, x_1, x_2, \ldots, x_t}[\sigma_{\vee_{1 \leqslant i \leqslant t}(x_i \theta d)}(B)]$, where *Com* is the set of common attributes of R and B.
- C : For each relation with relation name $C_i \in Dom(a)$ and C_i satisfies $P(a)$, pose the subquery $\sigma_{P(a)}(C_i)$.

Example 4.2. Consider the global selection $\sigma_{\text{index} \geqslant 10000}(\mathbf{A\_B\_C})$ posed on the global relation **A_B_C** in Fig. 8. Then, the subqueries for A, B and C are:

- $A : \sigma_{\text{index} \geqslant 10000}(A)$,
- $B : \pi_{\text{date,Tokyo,HongKong,Taipei,NewYork}}[\sigma_{\text{Tokyo} \geqslant 10000 \vee \text{HongKong} \geqslant 10000 \vee \text{Taipei} \geqslant 10000 \vee \text{NewYork} \geqslant 10000}(B)]$
- $C : \sigma_{\text{index} \geqslant 10000}(\text{Tokyo})$, $\sigma_{\text{index} \geqslant 10000}(\text{HongKong})$, and $\sigma_{\text{index} \geqslant 10000}(\text{Taipei})$.

(3) *The involved attribute is a non-versatile attribute.* In such circumstances, there may be no corresponding attributes in a local site (since the involved attribute is derived from other sites). The global site should detect such problems and send no subqueries to such local sites. However, if this problem does not occur, then the subqueries for A, B and C are:

- $A : \sigma_{P(a)}(A)$,
- $B : \sigma_{P(a)}(B)$,
- $C : \sigma_{P(a)}(C_i)$, for each relation C_i.

4.2. Query decomposition for projection

We consider projection operations involving a single attribute only. The decomposition also depends on the type of the involved attribute.

(1) *The involved attribute is a versatile attribute.* In such cases, the system catalogs in **B**'s and **C**'s sites should be employed. We assume that in a local site, the system catalog contains the following tables:

- **Systables.** This table stores related information about all user relations, including a relation's name, its owner, and its attribute count.
- **Syscolumns.** This contains relative information about all attributes, including an attribute's name, the corresponding relation name, and the data type. Suppose in Example 2.1 the system catalog in Site 2 contains the following information.

Systables

Name	Owner	Colcount
B	frank	5

Syscolumns

Name	Relation_name	Type
Date	B	date
Tokyo	B	integer
HongKong	B	integer
Taipei	B	integer
NewYork	B	integer

Then, for a global projection $\pi_a(R)$, where a is a versatile attribute, the subqueries for relations A, B and C are as follows:

- $A : \pi_a(A)$,
- $B : \pi_{\text{name}}[\sigma_{\text{relation\_name}=\text{'B'}}(\text{Syscolumns})]$,
- $C : \pi_{\text{name}}(\text{Systables})$.

(2) *The attribute is a description attribute.* For a global projection, $\pi_a(R)$, where a is a description attribute, then the subqueries for relations A, B and C are as follows:

- $A : \pi_a(A)$,
- $B : \pi_{x_1,x_2,\ldots,x_t}(B)$ if $Dom(a) \cap Sch(B) = \{x_1, x_2, \ldots, x_t\}$,
- $C : \pi_a(C_i)$, for all C_i.

Example 4.3. For the projection $\pi_{\text{index}}(A\_B\_C)$, the subqueries for A, B and C are:

- $A : \pi_{\text{index}}(A)$,
- $B : \pi_{\text{Tokyo,HongKong,Taipei,NewYork}}(B)$,
- $C : \pi_{\text{index}}(\text{Tokyo})$, $\pi_{\text{index}}(\text{HongKong})$ and $\pi_{\text{index}}(\text{Taipei})$.

(3) *The involved attribute is a non-versatile attribute.* In such circumstances, there may be no corresponding attributes in a local site. The global site should detect such problems and send no subqueries to such local sites. However, if this problem does not occur, then the subqueries for A, B and C are:

- A : $\pi_{P(a)}(A)$,
- B : $\pi_{P(a)}(B)$,
- C : $\pi_{P(a)}(C_i)$, for each relation C_i.

Example 4.4. For the projection $\pi_{\text{quantity}}(A\_B\_C)$, the subqueries for A, B and C are:

- A : No corresponding subquery,
- B : No corresponding subquery,
- C : \$ π_{quantity}(Tokyo), π_{quantity}(HongKong), π_{quantity}(Taipei).

4.3. Query decomposition for join

We restrict the join predicate P to involve a single attribute, say a, only. That is, a join in the following can be specified as $R \bowtie\!\!\lhd_{P(a)} S$, $P(a) \equiv a\theta d$, $\theta \in \{>, <, \geqslant, \leqslant, \neq, =\}$. The decomposition depends on whether the involved attribute is a versatile, a description, or a non-versatile one.

(1) *The involved attribute is a versatile attribute.* For a global join $R \bowtie\!\!\lhd_{P(a)} S$, $P \equiv a\theta d$, if a is a versatile attribute, then the subqueries for relations A, B and C are as follows:

- A : $R_A \bowtie\!\!\lhd_{P(a)} S_A$, where R_A and S_A are the participants for constructing the global relations R and S, respectively.
- B : $\sigma_{\text{true}}(R_B)$ and $\sigma_{\text{true}}(S_B)$, where R_B and S_B are the participants for constructing the global relations R and S, respectively.
- C : $\sigma_{\text{true}}(R_{C_i})$ and $\sigma_{\text{true}}(S_{C_i})$, for all R_{C_i} and S_{C_i} participating in constructing the global relations R and S, respectively.

Notice that, for Sites B and C, we have no corresponding join operation, since the involved attribute value is an attribute name in Site B and a relation name in Site C. Therefore, we transfer the whole relations participating in constructing the corresponding global relation from Sites B and C. These local relations will be respectively transformed and integrated into the corresponding global relation and then two global relations R and S are joined by predicate $P(a)$ to obtain the query answer. (Recall the five steps described in Section 4.)

(2) *The involved attribute is a description attribute.* For a global join $R \bowtie\!\!\lhd_{P(a)} S$, if a is a description attribute, then the subqueries for A, B and C are:

- A : $R_A \bowtie\!\!\lhd_{P(a)} S_A$, where R_A and S_A are the participants for constructing the global relations R and S, respectively.
- B : $\sigma_{\text{true}}(R_B)$ and $\sigma_{\text{true}}(S_B)$, where R_B and S_B are the participants for constructing the global relations R and S, respectively.
- C : $R_{C_i} \bowtie\!\!\lhd_{P(a)} S_{C_i}$, for all R_{C_i} and S_{C_i} participating in constructing the global relations R and S, respectively.

(3) *The involved attribute is a non-versatile attribute.* In such circumstances, there may be no corresponding attributes in a local site. The global site should detect such problems and send no subqueries to such local sites. However, if this problem does not occur, then the subqueries for A, B and C are:

- $A : R_A \bowtie\!\!\!\triangleleft_{P(a)} S_A$, where R_A and S_A are the participants for constructing the global relations R and S, respectively.
- $B : R_B \bowtie\!\!\!\triangleleft_{P(a)} S_B$, where R_B and S_B are the participants for constructing the global relations R and S, respectively.
- $C : R_{C_i} \bowtie\!\!\!\triangleleft_{P(a)} S_{C_i}$, for all R_{C_i} and S_{C_i} participating in constructing the global relations R and S, respectively.

5. Discussion and future work

The development of a heterogeneous database system requires the integration of data described by different schemas from multiple autonomous sources. This problem is further complicated if data are stored in heterogeneous databases with conflicting schema structures.

It has been pointed out in [20] that a multidatabase query language that supports users to pose queries on relations with conflicting schema structures is a second order logic. Our work provides another approach that integrates relations with conflicting schema structures. We propose to use *metadata* for resolving the semantic conflicts among autonomous systems. The metadata are prepared by each local database administrator and are reconciled by the global database administrator. We propose three integration procedures to integrate different types of structural conflict. By consulting the metadata, our approach transforms semantically related data and schemas (including attribute names and relation names) to a unified schema according to the type of structural conflict.

After integrating local schemas into a global one, users can pose queries on the global schema. However, the data of the integrated schema should be derived from each local site. Therefore, we also present the query decomposition rules for restricted selection, projection, and join operations to do global query optimization. We are currently studying more query decomposition rules for a general algebraic query operation.

Acknowledgements

The authors would like to acknowledge the anonymous referees whose invaluable comments and suggestions helped to improve this paper substantially. This research was supported by the National Science Council, Taiwan, ROC, under Contract No. NSC 86-2213-E-155-034.

References

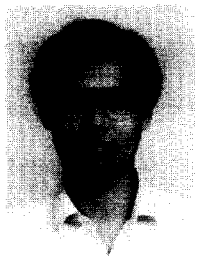
[1] A. Elmargamid, C. Pu (Eds.), *ACM Computing Surveys*, (Special Issue on Heterogeneous Databases) 22(3) (1990).

[2] S. Agarwal, A.M. Keller, G. Wiederhold, K. Saraswat, Flexible relation: An approach for integrating data from multiple, possibly inconsistent databases, *Proc. IEEE Int. Conf. on Data Engineering*, Taipei, Taiwan, 1995, pp. 495–504.

[3] C. Batini, M. Lenzerini, S.B. Navathe, A comparative analysis of methodologies for database schema integration, *ACM Computing Surveys* 18(4) (1986) 323–364.

[4] Y. Breitbart, P.L. Olson, G.R. Thompson, Database integration in a distributed heterogeneous database system, *Proc. IEEE Int. Conf. on Data Engineering* (1986) 301–310.

[5] Y. Breitbart, Multidatabase interoperability, *SIGMOD RECORD* 19(3) (1990) 53–60.
[6] O.A. Bukhres, A.K. Elmagarmid, *Object-oriented Multidatabase Systems: Solutions for Advanced Applications* (Prentice-Hall, 1996).
[7] A.L.P. Chen, J.S. Chiu, F.S.C. Tseng, Evaluating aggregate operations over imprecise data, *IEEE Trans. on Knowledge and Data Engineering* 8(2) (1996) 273–284.
[8] B. Czejdo, M. Rusinkiewicz, D.W. Embley, An approach to schema integration and query formulation in federated database systems, *Proc. IEEE Int. Conf. on Data Engineering* (1987) 477–484.
[9] C.J. Date, The outer join. In: S.M. Deen, P. Hammersley (Eds.): *Proc. 2nd Int. Conf. on Databases (ICOD-2)* (Wiley, 1983), pp. 76–106.
[10] U. Dayal, H.Y. Hwang, View definition and generalization for database integration in a multidatabase system, *IEEE Trans. on Software Engineering* 10(6) (1984) 628–644.
[11] S.M. Deen, R.R. Amin, M.C. Taylor, Data integration in distributed databases, *IEEE Trans. on Software Engineering* 13(7) (1987) 860–864.
[12] L.G. DeMichiel, Resolving database incompatibility: An approach to performing relational operations over mismatched domains, *IEEE Trans. on Knowledge and Data Engineering* 1(4) (1989) 485–493.
[13] J. Grant, W. Litwin, N. Roussopoulos, T. Sellis, Query languages for relational multidatabases, *The International Journal on Very Large Data Bases—The VLDB Journal* 2 (1993) 153–171.
[14] D. Heimbigner, D. McLeod, A federated architecture for information management, *ACM Trans. on Office Information Systems*, 3(3) (1985) 253–278.
[15] D. Hsiao, Tutorial on federated databases and systems (Part 1), *The Int. Journal on Very Large Data Bases—The VLDB Journal* 1(1) (1992) 127–179.
[16] D. Hsiao, Tutorial on federated databases and systems (Part 2), *The Int. Journal on Very Large Data Bases—The VLDB Journal* 1(2) (1992) 285–322.
[17] Special issue on heterogeneous distributed database systems, F.J. Maryanski and H.-J. Schek (Eds.), *IEEE Computer* 24(12) (1991).
[18] W. Kim, J. Seo, Classifying schematic and data heterogeneity in multidatabase systems, *IEEE Computer* 24(12) (1991) 12–18.
[19] W. Kim (ed.), *Modern Database Systems: The Object Model, Interoperability and Beyond*, Addison-Wesley (1995).
[20] R. Krishnamurthy, W. Litwin, W. Kent, Language features for interoperability of databases with schematic discrepancies, *Proc. ACM SIGMOD—Int. Conf. on Management of Data* (1991) 40–49.
[21] C. Lee, C.-J. Chen, H. Lu, An aspect of query optimization in multidatabase systems, *ACM SIGMOD RECORD* 24(3) (1995) 28–33.
[22] W. Litwin and A. Abdellatif, An overview of the multi-database manipulation language MDSL, *Proc. of the IEEE 75* (5) (1987) 621–632.
[23] W. Litwin, A. Abdellatif, B. Nicolas, Ph. Vigier, A. Zeronnal, MSQL: A multidatabase manipulation language, *Information Sciences: An International Journal* 49(1) (1987) 59–101.
[24] A. Motro, Superviews: Virtual integration of multiple databases, *IEEE Trans. on Software Engineering* 13(7) (1987) 785–798.
[25] M.P. Reddy, B.E. Prasad, P.G. Reddy, A. Gupta, A methodology for Integration of heterogeneous databases, *IEEE Trans. on Knowledge and Data Engineering* 6(6) (1994) 920–933.
[26] E. Sciore, M. Siegel, A. Rosenthal, Using semantic values to facilitate interoperability among heterogeneous information systems, *ACM Trans. on Database Systems* 19(2) (June, 1994) 254–290.
[27] M. Siegel, S.E. Madnick, A metadata approach to resolving semantic conflicts, *Proc. 17th International Conference on Very Large Data Bases (VLDB)*, (1991) 133–145.
[28] F.S.C. Tseng, A.L.P. Chen, W.P. Yang, Generalizing the division operation on indefinite databases, *Proc. 2nd Far-East Workshop on Future Database Systems*, Koyto, Japan (1992) 347–354.
[29] F.S.C. Tseng, A.L.P. Chen, W.P. Yang, Searching a minimal semantically equivalent dubset of a set of partial values, *The Int. Journal on Very Large Data Bases—The VLDB Journal* 2(4) (1993) 489–512.
[30] F.S.C. Tseng, A.L.P. Chen, W.P. Yang, Refining imprecise data by integrity constraints, *Data and Knowledge Engineering* 11(3) (1993) 299–316.
[31] F.S.C. Tseng, A.L.P. Chen, W.P. Yang, Answering heterogeneous database queries with degrees of uncertainty, *Distributed and Parallel Databases—An International Journal* 1(2) (1993) 281–302.



Frank S.C. Tseng received the B.S., M.S. and Ph.D. degrees, all in Computer Science and Information Engineering from the National Chiao Tung University, Taiwan, in 1986, 1988 and 1992, respectively. From 1993 to 1995, he served his military obligation both in the Chinese Air Force Academy (CAFA) and the General Headquarters of the ROC Air Force. Dr. Tseng was one of the winners of Acer Long Term Ph.D. dissertation prize in 1992. He joined the faculty of the Department of Information Management, Yuan-Ze University, Chung-Li, Taiwan in August, 1995. From 1996 to 1997, he was the Chairman of the Department of Information Management. Then he joined the faculty of the Department of Information Management, National Institute of Technology at Kaohsiung, Taiwan in August 1997. His current research interests include distributed database systems, schema integration in federated database systems, data warehousing and data mining. He has had papers published in The International Journal on Very Large Data Bases (the VLDB Journal), IEEE Transactions on Knowledge and Data Engineering, Data and Knowledge Engineering, Distributed and Parallel Databases: An International Journal, and Journal of Computer and Information Science. Dr. Tseng is a member of the IEEE Computer Society and the Association for Computing Machinery. He was listed in the 1997 International Who's Who of Information Technology.

Jeng-Jye Chiang received the M.S. degree in Computer and Information Science from the National Chiao Tung University in 1993. From 1993 to 1995, he served his military obligation in the General Headquarters of the ROC Air Force. He was a software engineer in GenNet Technology, Taiwan, from 1995 to 1996. Then he joined Origin (Taiwan) as a software engineer. From 1997, he was a senior technician in the Computer Center of the National Taiwan Normal University. His research interests include distributed database systems, network security and the Go game.

Wei-Pang Yang was born in May 17, 1950 in Hualien, Taiwan, Republic of China. Professor Yang received the B.S. degree in Mathematics from the National Taiwan Normal University in 1974, and the M.S. and Ph.D. degrees from the National Chiao Tung University in 1979 and 1984, respectively, both in Computer Engineering. Since August 1979, he has been on the faculty of the Department of Computer Science and Information Engineering at the National Chiao Tung University, Hsinchu, Taiwan. In the academic year 1985–1986, he was awarded the National Postdoctoral Research Fellowship and was a visiting scholar at Harvard University. From 1986 to 1987, he was the Director of the Computer Center of the National Chiao Tung University. In August 1988, he joined the Department of Computer and Information Science at the National Chiao Tung University, and acted as the Head of the Department for one year. Then he went to IBM Almaden Research Center in San Jose, California for another year as a visiting scientist. From 1990 to 1992, he was the Head of the Department of Computer and Information Science again. His research interests include database theory, database security, object-oriented databases, image databases and Chinese database retrieval systems. Dr. Yang is a senior member of IEEE, and a member of ACM. He was the winner of the 1988 and 1992 AceR Long Term Award for Outstanding M.S. Thesis Supervision, 1993 AceR Long Term Award for Outstanding Ph.D. Dissertation Supervision, and the winner of the 1990 Outstanding Paper Award of the Computer Society of the Republic of China. He also obtained the Outstanding Research Award of the National Science Council of the Republic of China.