DATA MANIPULATION IN HETEROGENEOUS DATABASES*

Abhirup Chatterjee and Arie Segev

Walter A. Haas School of Business
University of California at Berkeley
and
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
Berkeley, CA 94720

Abstract

Many important information systems applications require access to data stored in multiple heterogeneous databases. This paper examines a problem in inter - database data manipulation within a heterogeneous environment, where conventional techniques are no longer useful. To solve the problem, a broader definition for join operator is proposed. Also, a method to probabilistically estimate the accuracy of the join is discussed.

1 Introduction

The present information processing environment in large organizations can be characterized by a growing number of business applications that require accessing and manipulating data. The data is often located in *heterogeneous* hardware and software environments and distributed among the nodes of computer networks. The Database Management Systems (DBMSs) involved are heterogeneous because they use different underlying data models, different data definition and manipulation capabilities, and function in different operating environments. Data conveying the same information contained in heterogeneous data sources may have different physical and logical data representation and even different values [Bre90].

The objective of our ongoing research project is to analyze the problems of semantic discrepancy, incompatibility and heterogeneity in databases. As a step towards achieving the above objectives, we present in this paper a probabilistic technique for resolving data heterogeneity problems.

*Issued as LBL Technical Report 30870. This work was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

2 Data Heterogeneity

The data heterogeneity problem occurs due to incompatibility among similar attributes resulting in the same data being represented differently in different databases[1]. We distinguish between two types of incompatibility: structural and semantic. It should be noted, that for two attributes to be compatible, they need not share the same name.

Structural Incompatibility

Structural incompatibility occurs when the attributes are defined differently in different databases. Some of the sources of structural incompatibility are:

Type mismatch: The same attribute may have incompatible type definitions in different databases. For example, social security number could be of type 'character' in one database and 'numeric' in another. Similarly, an attribute may be set-valued in one database and single-valued in another.

Formats: Different databases often use different formats for the same data element, e.g., date in day/month/year versus month/day/year.

Units: Different databases use different units for the same data element. For instance, quantity of raw material may be expressed by the 'number of truck loads' or the total weight in tons or the dollar value.

Granularity: Data elements representing measurements differ in granularity levels, e.g., sales per month or annual sales.

Semantic Incompatibility

Semantic incompatibility occurs when similarly defined attributes take on different values in different

[1]This problem has been also referred to in the literature as the Instance Identification [WM89] or the Naming problem [BLN86].

databases. Some of the sources of semantic incompatibility are:

Synonyms: When the same entity is identified using different identifiers in different databases, the identifiers constitute synonyms. For example, an entity, IBM, may be identified as the 'International Business Machine' or 'IBM Corp' or simply as 'IBM' in different databases.

Homonyms: When different entities share the same identifier in different databases, they become homonyms. For example, a popular name like 'John Smith' may identify many persons.

Codes: Codes are used for various reasons, such as saving storage space. Codes are often local to the databases, and therefore non-uniform even when referring to the same domain.

Incomplete Information: Missing and incomplete information is represented by *null* values in relational databases. While some databases allow nulls, others do not. Moreover, the meaning of nulls (e.g., unknown, not applicable, unavailable) varies among databases.

Recording Errors: These could be due to typographical mistakes or variations in measurement. Typing errors happen frequently with similar sounding names, e.g., 'Smith', 'Schmidt' and 'Smythe'.

Surrogates: Surrogates are the system generated identifiers, used in different databases. They could have the same domain and meaning, but be otherwise unrelated.

Asynchronous Updates: These happen when data items, replicated in different databases, get updated at different points in time and become inconsistent. These are more likely if the data items are inherently time varying, such as a person's weight or age.

The definition of semantic incompatibility presented above is more restrictive than some of the definitions suggested in literature. For example, Sheth and Larson [SL90] defines semantic heterogeneity to include both the structural and semantic incompatibility, as defined in this paper. Sometimes it is difficult to decouple incompatibilities caused by differences in structures from those resulting from semantic differences. For example, the use of different codes may be considered by some as a structural difference.

We feel it is necessary to make a distinction between the two types of incompatibilities in our model. This is because, if the attributes are structurally incompatible, it is often meaningless to compare them directly, e.g., comparing weight in kilograms with that in pounds. In these cases, a transformation such as conversion of units, has to precede the comparison step. Semantic incompatibility, on the other hand, is harder to detect and resolve, e.g., no transformation could eliminate typographical errors.

The sources of heterogeneity listed above are not meant to be exhaustive. Other cases of heterogeneity are discussed in [DH84, BLN86]. As the relational data model is extended with newer data types, heterogeneity from other sources will have to be addressed. For example, using the subsets and cardinalities to compare the set-valued attributes. It is also possible to have combinations of different cases like synonyms and asynchronous updates occurring at the same time, which adds to the complexity of the problem.

3 Literature Review

A simple solution to the data heterogeneity problem could be to standardize the names. This is a viable option when, for example, the databases are small and/or autonomy is not crucial. But among autonomous databases, it will be extremely difficult to develop and practically enforce such comprehensive standards [Bre90].

The use of rules to resolve this problem has been suggested by Wang and Madnick in [WM89]. The rule based approach introduces more semantics to the solution. However, it is nontrivial to create and update the rule base. Secondly, the rule base may not be very portable across different environments. Further, rules require a detailed semantic knowledge of the underlying databases, which may not be available.

It has been suggested that one could store the identifiers of all possible synonyms of a particular object in a table and use it for conflict resolution [Mar91]. This is an ideal solution, which could be impractical for large databases.

Maybe tuples were used as qualitative measures of uncertainty while processing queries over incompatible domains [DeM89]. However, if there were any inconsistencies among the common attributes, the tuples were considered inconsistent and subsequently ignored. An information theoretic approach to model imprecise information in databases can be found in [Mor90]. Retrieval of multimedia documents using imprecise query specification is discussed in [RP90].

4 The Proposed Model - A Qualitative Introduction

In this paper, we present a new model for resolving the data heterogeneity problem. This model is based on probabilistic reasoning. It has certain advantages over conventional data manipulation methods as explained later. The model allows matching of records across databases when the identifying attributes (e.g., the keys) are structurally or semantically incompatible. We assume that a mapping step preceding the application of our technique will identify attributes in different databases which are compatible to each other.

In order to match entity instances in two relations, our model employs a special tactic. We compare not

only the identifying attributes as per the conventional methods, but *all* attributes which describe the entity instances and are common to the two relations[2]. This helps in the following way. Consider two tables which have structurally or semantically incompatible keys. Matching entity instances in these two tables could result in the following problems:

1. The two tables might use different identifiers to identify the same real world instance.

2. The tables might use the same identifier to identify different real world instances.

The conventional data manipulation operators will not be able to resolve either problem. In the first case, a straight forward comparison of the keys (if such comparison is possible[3]) will indicate that the entity instances they identify are different, even if they are the same. In the second case, the conventional operators will wrongly identify two different entities to be the same.

On the other hand, when *all* common attributes are compared according to our model, the potential for such errors is considerably reduced. In the first case, even if the keys are different, most of the other common attributes would match if two records describe the same real world instance. In the second case, if the two records sharing the same key refer to two different real world instances, the common attributes are less likely to match. Thus, by considering all common attributes, the probability of accurate identification is significantly improved. The idea of using non-key attributes to help identify tuples has been mentioned in literature [SG89].

After comparing a pair of records from two relations, a value, called the *comparison value* is assigned to the pair. This value measures how well one record matches the other record in the pair. A probabilistic model to estimate the comparison value is described in Section 6. The comparison value can also be used to rank the records for presentation to the user.

The above concepts are utilized to develop the theory of the Entity Operators (in short, E-operators). The E-join operator is discussed in detail in the next section along with an example. For a treatment on other data manipulation operators, viz., union, intersection, selection and duplicate elimination, the readers are referred to [CS91a].

5 The Entity Join

The Entity Join (in short, E-join) can be used to join records across different databases. A pair of tuples is selected, one each from the two relations being joined. (These relations could be from different databases.) The join attribute of the two records as well as other *useful* attributes which are common between the two relations are compared, if and when they are compatible. Depending on the number of matches between these attributes, a comparison value is assigned to the record pair. This value estimates the *correctness* in joining the records in the pair.

Let $r(R)$ and $r(S)$ be the two relations to be joined, where, R and S denote the schemas for the two relations respectively. Assume the join attribute to be a_J, where $a_J \in R, S$. Let the tuples of $r(R)$ be r_i, $i = 1, \ldots, K$. Similarly, let the tuples of $r(S)$ be denoted by s_i, $i = 1, \ldots, L$.

Let $\mathcal{M}$ be a set containing the result of E-joining $r(R)$ and $r(S)$ on a_J. Then $\mathcal{M}$ can be expressed as:

$$\begin{aligned} \mathcal{M} &= r(R) \bowtie r(S) \\ &= r(R) \times r(S) \\ &\quad \text{such that } r_i[a_J] \equiv s_j[a_J] \end{aligned}$$

The symbol "$\equiv$" is being used to indicate *equivalence*. Due to heterogeneity, the join attributes are often incompatible although they may be *equivalent*, i.e., may refer to the same object in real life. The cross product:

$$r(R) \times r(S) = \{(r_i, s_j): r_i \in r(R), s_j \in r(S), \forall \text{ i, j}\}$$

can now be expressed as the union of two disjoint sets:

$$\mathcal{M} = \{(r_i, s_j): r_i[a_J] \equiv s_j[a_J]; r_i \in r(R), s_j \in r(S)\}$$

and

$$\mathcal{U} = \{(r_i, s_j): r_i[a_J] \not\equiv s_j[a_J]; r_i \in r(R), s_j \in r(S)\}$$

The data heterogeneity problems introduced in Section 4 can now be mathematically formulated as follows:

1. for a given i, j, $r_i[a_J] \neq s_j[a_J]$ or $r_i[a_J]$ incomparable with $s_j[a_J]$ but $(r_i, s_j) \in \mathcal{M}$.

2. $r_i[a_J] = s_j[a_J]$ but $(r_i, s_j) \in \mathcal{U}$ for a particular i, j pair.

It is important to identify the set of *useful* common attributes in the two relations R and S that can be used to compute the Entity join. The identification of *useful* attributes depends on (1) whether the join attribute(s) are keys in their respective tables, (2) the cardinality/informativeness of the attribute, and, (3) the cost of including an additional attribute compared to the gain in accuracy. The readers are referred to [CS91b] for further discussion on this issue.

Entity joins can be computed over a wider range of conditions than those of a conventional join. E-join allows the join attributes to be structurally and semantically incompatible. Even when conventional join is possible, E-join is recommended if the existence of recording errors or asynchronous updates is suspected.

Example. A company wants to create a list of customers with good credit rating by joining its CUSTOMER table with the CREDIT table obtained from the Credit Bureau. The tables have the following

[2]We assume that the relations are from two different databases. The proposed model can also be applied to relations from the same database.

[3]In some cases of structural incompatibility, a direct comparison is not possible, e.g., type mismatch.

schema:
CUSTOMER = {**Customer Number**, Last Name, First Name, Street Address, City, State, Zip, Total Purchase Year to Date, Date last purchased}
CREDIT = {**Social Security Number**, Last Name, First Name, Street Address, City, State, Zip, Credit Rating}.

Note that the identifiers used in the two tables are different (the identifiers are in **bold** type). We assume that no inconsistencies exist among the common attributes and all of them can be meaningfully compared. Thus, the useful common attributes are:
CUSTOMER ∩ CREDIT =
{Last Name, First Name, Street Address, City, State, Zip}.

Consider a record from the CUSTOMER relation:
r =
(_, Smith, John, 51st Street, New York, NY, 10006, _, _)
and the record of the same person from the CREDIT relation, s =
(_, Smith, Jorn, 51st Street, New York, NY, 10006, _).
In these records only the common attributes are shown. The other attributes are irrelevant for the example.

A conventional join on last name-first name combination will not be able to match these two records as the first name is misspelled as "Jorn" in s. A join on last name alone will not be very useful as record r will get joined to all CREDIT tuples which have "Smith" as the last name. So, we need to perform an E-join in this case. (The example is continued in the next section.) □

6 Comparison Value Estimation

The result of comparing the common attributes of a pair of tuples is stored as the comparison value for the pair. A technique for qualitatively estimating the comparison value is given in [CS91a]. For applications requiring more precise estimation, a quantitative framework to calculate the comparison value is discussed below.

Let the useful attributes common between R and S be $\{a_j, j = 1, \ldots, n\} \subseteq \{R \cap S\}$. Let $t = (r_i, s_j)$: $r_i \in r(R)$, $s_j \in r(S)$. Let us define a vector as $\gamma(t) = \{\gamma_1(t), \gamma_2(t), \ldots, \gamma_n(t)\}$, where the number of components of $\gamma(t)$ is equal to the number of common attributes between R and S [FS69]. (We denote vectors in **bold** type.) The result of comparison of the two tuples r_i and s_j is stored in this vector.

The $\gamma(t)$ function can be defined in various ways. Since each component refer to a specific common attribute, different weights can be attached to it based on the informativeness of the attribute. Similarly, $\gamma_j(t)$ can be made to take on continuous values over a range depending on how close the values of a_j are in the two records. For the time, let us assume a binary vector which assigns equal weights to all the useful common attributes. Thus,

$$\begin{aligned}
\gamma_1(r_i, s_k) &= 1 \quad \text{if } r_i.a_1 = s_k.a_1 \\
&= 0 \quad \text{otherwise} \\
\gamma_2(r_i, s_k) &= 1 \quad \text{if } r_i.a_2 = s_k.a_2 \\
&= 0 \quad \text{otherwise} \\
&\vdots \\
\gamma_n(r_i, s_k) &= 1 \quad \text{if } r_i.a_n = s_k.a_n \\
&= 0 \quad \text{otherwise}
\end{aligned}$$

The comparison value can now be defined as a function of this vector, $CV(t) = f[\gamma(t)]$. The users might want to define functions of their choice. We present here a probabilistic model to estimate $CV(t)$. Since the comparison values are probabilities in our model, we denote them as tuple probabilities, $p_{tuple}(t)$.

Definition. Tuple Probability. Given the results of comparing the useful common attributes, the conditional probability that the join attributes are correctly matched. Formally,

$$p_{tuple}(t) = \Pr\{\, t \in \mathcal{M} \mid \gamma(t) \,\}.$$

We say that $t \in \mathcal{M}$ with certainty, when we have a 'perfect match', i.e., the comparison vector is a unit vector. Mathematically,

$$\Pr\{\, t \in \mathcal{M} \mid \gamma(t) = \mathbf{1} \,\} = 1.$$

Similarly, we say that $t \notin \mathcal{M}$ with certainty, when we have a 'perfect mismatch', i.e., the comparison vector is zero. Formally,

$$\Pr\{\, t \in \mathcal{M} \mid \gamma(t) = 0 \,\} = 0.$$

In order to estimate $p_{tuple}(t)$, we use the Bayes' Theorem.

$$\begin{aligned}
p_{tuple}(t) &= \Pr\{t \in \mathcal{M} \mid \gamma(t)\} \\
&= \frac{\Pr[t \in \mathcal{M},\ \gamma(t)]}{\Pr[\gamma(t)]} \\
&= \frac{\Pr[\gamma(t) \mid t \in \mathcal{M}].\Pr[t \in \mathcal{M}]}{\Pr[\gamma(t)]}
\end{aligned}$$

Thus, to evaluate the tuple probability, we need to estimate the two unconditional probabilities $\Pr\{\gamma(t)\}$ and $\Pr\{t \in \mathcal{M}\}$ and a conditional one, $\Pr\{\gamma(t) \mid t \in \mathcal{M}\}$.

The tuple probabilities can be computed most accurately if one has the knowledge of the set $\mathcal{M}$ and the above distributions. Typically, however, these would be some general distributions whose properties may not be available. For such situations, we use an algorithm which utilizes the recursive definition of tuple probability. Starting with some initial estimate of $\mathcal{M}$, which gets updated through successive iterations, the algorithm executes until there are no changes across two iterations [CS91a].

Example. (Continued from Section 5.) Comparison of the two records, r and s, results in the following comparison vector, having six components, one for each field in common:

$$\gamma(r,\ s) = [1,\ 0,\ 1,\ 1,\ 1,\ 1].$$

There is a zero in the second position as the first name in the two records do not match. Using this information, one can now estimate the comparison value:

$$\begin{aligned} CV(r,s) &= p_{tuple}(r,s) \\ &= \Pr\{(r,s) \in \mathcal{M} | \gamma(r,s) = [1,0,1,1,1,1]\} \end{aligned}$$

This calculation however is beyond the scope of the current paper. Qualitatively, it can be said that the probability will be high as five of six attributes match, implying that r and s most likely refer to the same individual. The conventional approach could not have made the inference with the same level of confidence as the probabilistic one. □

7 Conclusions

The heterogeneous environment will be a prevalent data processing environment for the next decade. This paper considers the problem of inter - database data manipulation in a heterogeneous environment. The problem arises due to the presence of heterogeneity among data values caused mainly by the users who name entities independent of each other in different databases.

In this paper, we proposed broader definitions for data manipulation operators. We discussed the E-join which allows joins to occur across mismatched, incompatible domains. A probabilistic model was presented for estimating the accuracy of the join in a heterogeneous environment.

References

[BLN86] M. Batini, C. Lenzirini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computer Surveys*, 18(4):323–363, December 1986.

[Bre90] Yuri Breitbart. Multidatabase interoperability. *ACM SIGMOD Record*, 19(3):53–60, September 1990.

[CS91a] Abhirup Chatterjee and Arie Segev. Information retrieval in heterogeneous databases. Technical Report 31117, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, University of California at Berkeley, Berkeley, CA 94720, 1991.

[CS91b] Abhirup Chatterjee and Arie Segev. A probabilistic approach to joins in heterogeneous databases. Technical Report 30754, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, University of California at Berkeley, Berkeley, CA 94720, 1991.

[DeM89] Linda G. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):485–493, December 1989.

[DH84] Umeshwar Dayal and Hai-Yann Hwang. View definition and generalization for database integration in a multidatabase system. *IEEE Transactions on Software Engineering*, SE-10(6):628–645, November 1984.

[FS69] I. P. Fellegi and A. B. Sunter. A theory of record linkage. *Journal of the American Statistical Association*, 64:1183–1210, December 1969.

[Mar91] Victor M. Markowitz. An architecture for identifying objects in multidatabases. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan*, pages 294–301. Sponsored by IEEE Computer Society and The Information Processing Society of Japan, April 1991.

[Mor90] J. M. Morrissey. Imprecise information and uncertainty in information systems. *ACM Transactions on Information Systems*, 8(2):159–180, April 1990.

[RP90] F. Rabitti and Savino P. Retrieval of multimedia documents by imprecise query specification. In *Proceedings of the International Conference on Extending Database Technology, Venice, Italy*, pages 203–218. Springer-Verlag, 1990.

[SG89] Amit P. Sheth and Sunit K. Gala. Attribute relationships: An impediment in automating schema integration. In *Proceedings of the Workshop on Heterogeneous Databases.* Sponsored by NSF, December 1989.

[SL90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computer Surveys*, 22(3):183–235, September 1990.

[WM89] Y. Richard Wang and Stuart E. Madnick. The inter-database instance identification problem in integrating autonomous systems. In *Proceedings of the Fifth International Conference on Data Engineering, Los Angeles, California*, pages 46–55, February 1989.