LBL--31117

DE92 016993

A Probabilistic Approach to Information Retrieval in Heterogeneous Databases

Abhriup Chatterjee and Arie Segev

Walter A. Haas School of Business
University of California at Berkeley

and

Information and Computing Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

August, 1991

Presented at the Workshop on Information Technologies & Systems
December 14-15, 1991, Cambridge, MA

MASTER

This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract DE-AC03-76SF00098.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

A PROBABILISTIC APPROACH TO INFORMATION RETRIEVAL 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

During the past decade, organizations have increased their scope and operations beyond their traditional geographic boundaries. At the same time, they have adopted heterogeneous and incompatible information systems independent of each other without a careful consideration that one day they may need to be integrated. As a result of this diversity, many important business applications today require access to data stored in multiple autonomous databases.

This paper examines a problem of inter - database information retrieval in a heterogeneous environment, where conventional techniques are no longer efficient. To solve the problem, broader definitions for join, union, intersection and selection operators are proposed. Also, a probabilistic method to specify the selectivity of these operators is discussed. An algorithm to compute these probabilities is provided in pseudocode.

*Issued as LBL Technical Report 31117. 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.

Contents

1 Introduction 1
- 1.1 Heterogeneity in environment . . . 1
- 1.2 Database Integration . . . 1
- 1.3 Contributions of the paper . . . 1
- 1.4 Applications . . . 2
- 1.5 Organization of the paper . . . 2

2 Data Heterogeneity 3

3 Literature Review 4
- 3.1 General Research Issues . . . 4
- 3.2 Data Heterogeneity Resolution Methods . . . 5

4 The Proposed Model - A Qualitative Introduction 6

5 The Entity Join 7

6 The Entity Union and Entity Intersection 8

7 The Entity Selection 9

8 Comparison Value Estimation 10
- 8.1 Union, Intersection and Join . . . 10
- 8.2 Selection . . . 11
- 8.3 Definition of Tuple Probability . . . 11

9 The Threshold Probability and Decision Rule 12

10 An Algorithm to Estimate Tuple Probabilities 13

11 Conclusions 14

1 INTRODUCTION

1.1 Heterogeneity in environment

During the past decade, organizations have increased their scope and operations beyond their traditional geographic boundaries. In order to survive the stiff market competition, the number of mergers, joint ventures and takeovers both within and across national boundaries have increased at a tremendous rate. At the same time, significant advances in technology have provided opportunities for dramatically increasing the number, type, size and complexity of the information systems. The organizations had initially adopted these diverse and incompatible systems in an uncoordinated way, independent of each other without a careful consideration that one day they may need to be integrated. As a result of such a diversity in existing information systems, many important applications in the 90's will require access to multiple disparate information systems both within and across organizational boundaries.

The present information processing environment in large organizations can be characterized by a growing number of business applications that require accessing and manipulating data from various preexisting, autonomous databases. These databases are 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 logical and physical representation and even different values [Bre90].

1.2 Database Integration

The objective of our research, in a broad sense, is to develop techniques that will provide the user with a uniform or *integrated* view of the data in heterogeneous databases. In general, such integration can be achieved in two ways:

Physical Integration: A single large database physically replaces all preexisting databases, i.e., no heterogeneity is allowed in any way.

Virtual Integration: Such integration creates an illusion of a single database system and hides from the users the intricacies of different DBMSs and access methods, without imposing any restrictions on the individual databases.

The latter approach generated significant interest in the database research community. Physical conversion of large, independently managed databases to a common, globally acceptable model may be infeasible not only due to the huge time and monetary investment required, but also because of the lack of hardware, software, and technical staff support. Some of the users will have to learn a new system, which could be inconvenient as well. Because of the large overhead associated with physical integration, it is predicted that most organizations will opt for virtual integration of their information systems.

Efficient virtual integration of heterogeneous databases requires the solution of the following problems: (1) Schema integration, (2) Data heterogeneity, (3) Query optimization, (4) Transaction management, and (5) Object-orientation in heterogeneous environment. A review of heterogeneous database literature shows that considerable amount of progress has been made in Schema integration (ref Section 3.1). We believe that there are significant opportunities of further research in the remaining areas. The focus of this paper will be on resolving data heterogeneity problems.

1.3 Contributions of the paper

Probabilistic information retrieval techniques developed in this paper can be applied in many settings. Most of these application areas deal with medium to large data sets managed by special or general purpose database management systems. Traditionally, the data heterogeneity issues used to be clerically resolved. However, as many of these problems occur repeatedly, clerical intervention became too costly, unreproducible, error-prone and time consuming to be a viable option [Jar89]. As a result, researchers in various fields computerized the resolution process and emerged with several special purpose solutions which they

incorporated in their systems. Our goal is to provide a general purpose solution to these problems. To this end, we accomplish the following in this paper:

- Identify and define the basic concepts and taxonomy of data heterogeneity problems
- Develop representational models that facilitate the resolution of discrepancy, heterogeneity and incompatibility among the data.
- Formulate probabilistic techniques for identification and retrieval of necessary information from various databases using incomplete and insufficient knowledge.
- Design algorithms and control parameters to provide users with the flexibility of specifying data accuracy requirements.

1.4 Applications

It is our observation that data heterogeneity problems occur in many settings and the proposed information retrieval technique will have numerous applications in business, social, biological and physical sciences. In this section, we list a number of examples which will be directly benefited from this approach. The applications can be classified into three broad classes depending on the nature of the underlying problem one is trying to solve. These classes are as follows:

Approximate Matching of Common Objects: In this class of application, the objective is to identify records pertaining to the same object from multiple databases. Examples of such applications are: frame creation in U.S. census [CB88], coverage estimation in surveys [Key79], long term medical follow up studies in epidemiology [CF90], immigration control [CH90] and forensics [Taf70].

Finding Similar Objects: In this class of application, the objective is to find other distinct objects from same/different databases which have the similar characteristics as the test object. The problem in this case is of identification as the name or the identification number of the object being retrieved is not be known in advance. So the retrieval is based on the similarity of other attributes between the test and the retrieved objects. Applications in this category are: document matching [SM83], comparison of chemical properties [JM90], cluster analysis [Lor83] and matched pair sampling [Ros89].

Classification by nearest neighbor: The construction of taxonomy is a fundamental undertaking in science. This class of application is an extension to the one discussed above. The objective is not only to find objects (or groups of objects) similar to the test object but also to assign the test object to a group based on these similarities. Examples belonging to this category can be found in biology [SGJ86] and political science [MW64].

We believe that most of these applications will be directly benefited from this research. These issues are discussed in detail in [CS91].

The ultimate business value of our research is to help organizations achieve competitive advantage through superior database management techniques. Higher precision in information retrieval involves higher cost. The model proposed in this paper provides sufficient flexibility to the users to strike a balance between cost and accuracy. Thus, for applications where precision is crucial, the model parameters could be adjusted to meet the application/user specifications. Techniques derived in this research can also be built into a management Decision Support environment.

While the problem of data heterogeneity is likely to be more pronounced in a heterogeneous environment, it could also occur within a single database. For example, the data pertaining to the same object can be entered differently by the different users in a single database. The results presented in this paper can be effectively used in such a situation. The current commercial systems do not provide much safeguard against this situation. They mostly leave it up to the user and/or the Database Administrator to ensure that the data representation is consistent and standard across the database.

1.5 Organization of the paper

The following is the organization of the paper. In Section 2, the data heterogeneity problem is discussed in detail. In Section 3, the research that has been done in this area is reviewed. Section 4 presents a

qualitative introduction to our model. In Sections 5-7, new operators are defined for inter-database join, union, intersection and selection operations. The issue of estimating the *Comparison Value* is discussed in Section 8. The concept of threshold probability is discussed in Section 9. In Section 10, an algorithm to estimate the probabilities is presented. The paper is concluded in Section 11 with a summary and directions for future research.

2 Data Heterogeneity

In order to process queries in a heterogeneous environment, attributes of a relation in one database often needs to be compared with the attributes of another relation in another database. Conventional operators require that such comparisons be done between compatible attributes. Considerable amount of research on establishing compatibility or equivalence between attributes has been reported in literature [LNE89, SG89, SSG+91]. A simple definition of compatibility for the purpose of this paper is given below.

Definition 1 *Compatibility. Let $dom(A)$ and $dom(B)$ be the domains of attributes A and B respectively. Then, a necessary condition for compatibility of A and B is*

$$dom(A) \cap dom(B) \neq \emptyset.$$

Compatibility does not necessarily require the attributes involved to have identical domains or names. However, for the comparison of A and B to be meaningful, they need to have the same semantics.

For example, names and numbers are not compatible and hence cannot be compared. This is because their domains have an empty intersection although a number and a name may refer to the same object in real life.

Data heterogeneity problems occur 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*.

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 structured attributes take on different semantics and values in different databases. Some of the sources of semantic incompatibility are as follows:

[1] This problem has been also referred to in the literature as the Instance Identification [WM89] or the Key Equivalence problem [Pu91].

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. This is so because, 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 in order to make the attributes structurally compatible. 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, BOT86]. 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

In this section, we first review some of the research done in heterogeneous databases in the context of the problems identified in Section 1.2. We then discuss in detail the solutions to the data heterogeneity problems proposed in literature.

3.1 General Research Issues

In this section, the major areas of current research in virtual integration of heterogeneous databases are briefly described. Most of the recent work has been concentrated in the following areas: Schema integration, Transaction management, Query optimization, and Object-orientation in heterogeneous databases.

Schema Integration

Schema integration involves creation of an integrated schema for a given set of local database schemas in such a way that each local schema can be considered as a view of the integrated schema. Both physical and virtual integration of heterogeneous databases (discussed in Section 1.2) require integration of schemas. Many approaches and techniques for schema integration have been reported in literature. A comprehensive comparison of twelve such integration methodologies is provided by [BLN86].

Transaction Management

The major task of transaction management in a heterogeneous environment is to ensure global consistency and freedom from deadlocks in the presence of local transactions. The problem has been extensively studied in two basic directions: restricted autonomy [Pu87, Pu88] and complete preservation of local autonomy [AGMS87, DE89, BS88]. However, there is no satisfactory algorithm where no restriction is imposed on the local DBMS. All the algorithms proposed so far either impose a restriction on the type of global transaction or assume the structure of the local concurrency control mechanism.

Query Optimization

Query optimization in heterogeneous systems was first addressed in Multibase [LR82]. It was done in two steps: Global and Local. During the global optimization step, the query was subdivided into various sub-queries which were then sent to the different local sites for processing. The local sites then locally optimized the sub-queries that were allocated to them. Many heterogeneous prototype systems include a query optimizer but not many have been described. At this time the research in this area is at a very early stage.

Object Orientation

Research in object-oriented heterogeneous databases has just started and a few papers have appeared so far [BNPS89]. Several authors ([Kim89, Bre90]) believe that potentially object oriented systems may be very important and contribute to the solution of domain mismatch, transaction management and query optimization problems. However, much more research is needed to evaluate the benefits of this approach.

3.2 Data Heterogeneity Resolution Methods

In this section, we review the solutions to data heterogeneity problems proposed in literature. A simple approach could be to make the system prompt the user, using some triggering mechanism, each time there is a conflict [DH84]. Clearly, such a system will take a fairly long time to process queries which makes it impractical. Another possible solution 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].

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, but for large databases, this could be impractical since (1) this table could get very large and have to be duplicated, and, (2) referential integrity rules have to be adjusted. Alternately, one could remane the entities in case of conflicts [BOT86].

The use of rules to resolve this problem has been suggested by Wang and Madnick in [WM89]. This approach is similar to ours in the sense that the non-unique attributes are used to identify instance. The rule based approach introduces more semantics to the solution. However, this approach does not model the uncertainty in the identification process in any way. Secondly, the rule bases are nontrivial to create and maintain and may be too specific to be portable across different applications. Further, rules require a detailed semantic knowledge of the underlying databases, which may not be available.

A qualitative probabilistic approach to this problem has been suggested in [Pu91]. This approach requires creation of a table of all possible values of an attribute (e.g., different spellings of names) to identify instances. Thus, this approach is somewhat similar to the one suggested in [Mar91] except for the use of probabilities.

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 [RS90].

4 The Proposed Model - A Qualitative Introduction

In this paper, we present a probabilistic model for resolving the data heterogeneity problem. 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 the steps 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[2], 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[3]. 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[4]) 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 [WM89, SG89].

There is some uncertainty associated with the identification of instances in this model from the possibility of a wrong match. The uncertainty exists because it is difficult to determine for sure if two records identify the same real world instance. A probabilistic approach is used in this paper to model the uncertainty. 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[5]. This 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 significance of "Entity" is that we are interested in identifying the tuples that refer to the same entity instances in different databases. The E-join operator is discussed in detail in the next section. Other data manipulation operators are developed in the subsequent sections.

The advantages of this model over the existing approaches are:

- it reduces explicit user involvement in query evaluation and consequently reduces the response time
- it saves communication and storage costs by not having to store and access large amounts of data on synonyms
- it provides a unified treatment of a number of semantic incompatibility issues, and
- it provides an estimate of the accuracy of matching by modeling the uncertainty in a natural way.

[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] This is true for union, intersection and join operators. The case of selection is slightly different and is discussed in Section 7.

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

[5] The meaning of comparison value in the context of selection is explained later.

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 accurate result of 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:

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

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

$$\begin{aligned} \mathcal{M} &= \{(r_i, s_j) : r_i[a_J] \equiv s_j[a_J];\ r_i \in r(R),\ s_j \in r(S)\} \\ \text{and} & \\ \mathcal{U} &= \{(r_i, s_j) : r_i[a_J] \not\equiv s_j[a_J];\ r_i \in r(R),\ s_j \in r(S)\} \end{aligned}$$

Our ultimate objective is to estimate $\mathcal{M}$ and hence the result of the join. This would require resolution of the data heterogeneity problems introduced in Section 4. These problems 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 influence of these factors on the usefulness of an attribute is under further research.

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.

It may be noted that when there are no attributes common between the joining relations other than the join attribute, the E-join defaults to the conventional join. A conventional join is thus a *trivial* case of the E-join. Unless specifically mentioned, all future reference to the E-join would refer to the non-trivial case.

Example 1. 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 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 section 8.) □

6 The Entity Union and Entity Intersection

In this section, two important relational database operators, the Entity Union and the Entity Intersection, are introduced. The reason for treating these two operators together is that the result of an intersection of two relations is a subset of their union. Thus, issues related to the union apply to the intersection as well.

The conventional union (or intersection) of two relations requires the two to be *union compatible*. Union compatibility can defined as follows [Dat90]:

Definition 2 *Union Compatibility. Two relations are union compatible if they are of the same degree, n, and the ith attribute of each (i = 1, 2, ..., n) share the same domain. Note that this does not mean that they need to have the same name.*

The union compatibility is imposed to guarantee the closure property, that is, to ensure that the result of union (or intersection) is still a relation and not a heterogeneous collection of dissimilar tuples.

The result of a union of two relations is the set of tuples present in either or both the relations. Using relational calculus, union can be defined as follows:

Let two relations be $r(XZ)$ and $s(XZ)$, where X is the key and Z the set of non-key attributes. The result of a union of $r(XZ)$ and $s(XZ)$,

$$\begin{aligned} r \cup s = \{\ t \ \mid\ & ((t \in r \wedge (\nexists u)(u \in s \wedge (u[X] = t[X]))) \\ & \vee (t \in s \wedge (\nexists u)(u \in r \wedge (u[X] = t[X]))) \\ & \vee (\exists u, v)(u \in r \wedge v \in s \wedge (t[X] = u[X] = v[X]) \wedge (\forall C)(C \in Z \wedge (t[C] = u[C] = v[C]))))\}. \end{aligned}$$

The intersection of the two relations results in the retrieval of those tuples which are present in both the relations. In terms of relational calculus, this can be defined as

$$r \cap s = \{t \mid ((\exists u, v)(u \in r \wedge v \in s \wedge (t[X] = u[X] = v[X]) \wedge (\exists C)(C \in Z \wedge (t[C] = u[C] = v[C]))))\}.$$

The relations r and s are inconsistent if

$$(\exists u, v)(u \in r \wedge v \in s \wedge (u[X] = v[X]) \wedge (\exists C)(C \in Z \wedge (u[C] \neq v[C]))).$$

This is clearly a strict condition for union and intersection. In a heterogeneous environment, when the two relations are from two independently managed databases, they are very likely to be inconsistent (according to the above definition of inconsistency). This could be due to the existence of structural and/or

semantic incompatibility between their key and other common attributes. Thus, its clear that conventional union (or intersection) may not be possible in such situations though such operations may be crucial for many applications.

Example 2. Consider two mailing list relations, $r(R)$ and $s(R)$ having the same schema, R = {Name, Address, Total-purchase-year-to-date, Most-recent-activity}. Consider a tuple of $r(R)$, r_1 = {John Smith, Sacramento, 465, 7/6/91}. The tuple corresponding to the same person in $s(R)$, s_1 = {J. Smith, Sacramento, 325, 3/3/91}. The Total-purchase-year-to-date is different in the two tables because $s(R)$ does not record the latest purchase John made. Also, $s(R)$ uses an abbreviated form of the name. (Note that John Smith and J. Smith are synonyms.) Ideally, an intersection of the two tables should identify John's record as present in both tables. Unfortunately, the conventional intersection will not be able do so because of the mismatched key and an inconsistent common attribute. □

In order to union and intersect tables across different databases, the Entity Union and Entity Intersection are defined. These are abbreviated to E-Union and E-Intersection respectively. These operators allow the attributes of the common tuples to be incompatible across the two relations. To compute the union (or intersection), the key attributes are compared. In addition, the values of the other common attributes are compared as well. It should be noted that all attributes common to the two relations belong to the useful intersection of R and S, unless there is some incompatibility between them.

Let $\mathcal{M}$ be a set containing the accurate result of intersection of $r(XZ)$ and $s(XZ)$. Then, $\mathcal{M}$ can be expressed as:

$$\mathcal{M} = \{t : t[X] = r_i[X] \equiv s_j[X]\}.$$

As before, the symbol "$\equiv$" is used to mean *equivalence*. Our objective is to estimate $\mathcal{M}$, the result of the intersection. This may not be easy due to data heterogeneity, as the keys may not be identical although they may be equivalent, i.e., they may refer to the same entity in real life. Synonyms can cause $r_i[X] \neq s_j[X]$ for a given i, j, though the corresponding tuples may belong to $\mathcal{M}$. Similarly, homonyms can lead to $r_i[X] = s_j[X]$, when the tuples do not belong to $\mathcal{M}$.

In order to estimate $\mathcal{M}$, pairs of records are selected, taking one from each of the two tables being unioned (or intersected). Depending on the number of matches among the common attributes, a comparison value is assigned to the pair. This value plays the key role in determining if a record belongs to $\mathcal{M}$. The result of intersection contains only the tuples that are present in both the relations. So if the comparison value is high, it is indicative of the fact that the tuple exits in both the relations and should be included in $\mathcal{M}$, the result of intersection. If, on the other hand, this value is low or zero, then the tuples being paired actually refer to different entities. For such tuples, no corresponding tuples exist in the other relation. Such tuples are to be ignored during intersection; and included in $\mathcal{U}$ as distinct tuples during a union. The result of E-union is then the union of $\mathcal{U}$ and $\mathcal{M}$.

Thus E-union and E-intersection allow union and intersection to be computed over broader ranges of situations. Even if the same entity has been identified differently in different databases or if some of the common attributes of common tuples are inconsistent due asynchronous updates, the E-operators will be able to identify them correctly to a large extent.

7 The Entity Selection

In this section, the problem of *selection* in a heterogeneous environment is analyzed. Using the notation of relational calculus, conventional selection can be defined as follows:

Let $r(XC)$ be a relation, where X is the key and C, a single attribute. Then, the result of selecting tuples for which attribute $C = z$ is

$$\sigma_{X = x} r(XC) = \{t \mid t \in r \wedge (t[X] = x)\}.$$

This, again, is a strict condition for selection. The constraint $t[C] = z$, assumes that the user knows exactly what (s)he is looking for. In a heterogeneous environment, the users often have only partial information, particularly when they are querying other databases. This could be due to the general unfamiliarity with a foreign database, or the presence of synonyms, homonyms, unfamiliar formats, codes and structures. Also,

independently managed databases can get asynchronously updated at any time, making even the duplicated tuples inconsistent. In such situations, strictly conventional selection may not be adequate to capture the necessary data.

In order to retrieve records from different databases, *Entity Select*, which we shall abbreviate to E-select, has been defined. Let $\mathcal{M}$ be a set containing the accurate result of selection. Then, $\mathcal{M}$ can be mathematically expressed as:

$$\mathcal{M} = \sigma_{X = x} r(XC) = \{t \mid t \in r \wedge (t[X] \equiv x)\}.$$

Our objective is to estimate $\mathcal{M}$, the result of the selection. We define two types of selection conditions: ***primary*** and ***secondary***. The constraints like $t[X] = x$ are called the primary conditions. The users are most interested in the tuples that satisfy the primary conditions. However, due to heterogeneity, none of the tuples may satisfy the primary conditions. In such situations, it is often worthwhile to consider the records which satisfy the secondary conditions. The secondary conditions are the additional filter conditions obtained from the user, which are expected to be true for most of the tuples selected by the primary conditions. For a primary condition $X = x$ and a secondary condition $C = z$, one may compute $\mathcal{M}$ as:

$$\begin{aligned} \sigma_{X = x \wedge C = z} r(XC) = \{t \mid t \in r \;\; & \wedge \;\; ((t[C] = z \wedge t[X] = x) \\ & \vee \;\; (t[C] = z \wedge t[X] \neq x) \\ & \vee \;\; (t[C] \neq z \wedge t[X] = x))\}. \end{aligned}$$

The E-select assumes a wider selection condition than its corresponding conventional counterpart. It will always retrieve the tuples which match the primary criteria. In addition, it also retrieves the tuples which satisfy the secondary conditions. The result of an E-select is thus a superset of the result of a conventional select. The power of the E-select comes from the fact that it provides a lot more flexibility and selectivity to a query.

A *comparison value* is assigned to each selected tuple depending on the number of selection conditions it satisfies. This value indicates the accuracy of selection, that is, to what extent the selected tuple satisfies the selection criterion. If this value is high, it means the corresponding tuple satisfies a large number of conditions. A low value indicates otherwise. Of course, for the E-select to be useful and non trivial, there must be at least one selection condition. Otherwise, E-select will default to a conventional one. This is because, if there are no conditions, the issue of satisfying the selection criteria does not arise. The section is concluded with an interesting application of E-select.

Example 3. Consider the problem of key word searching in a library, where the objective is to retrieve the documents which contain all the specified keywords. If this is posed as a conjunctive query, often no records are retrieved. This is because, there may not be any record in the database which contains *all* the keywords, or, the documents might be using synonyms of the requested keywords. Clearly, posing the query as a disjunctive one is not preferable, as far too many records may be retrieved. An E-selection is very useful in this situation. The important keywords may be requested in the primary conditions. Other preferred keywords may be requested in the secondary conditions. Even if there may not be any document which contains all the keywords, E-select will retrieve the documents which contains some of them. Further, the documents may be ordered by their selection value, so that the documents which best satisfy the selection criteria occur at the top of the output file. □

8 Comparison Value Estimation

In this section, a probabilistic framework to estimate the comparison value is presented. Since the treatment of union, intersection and join is slightly different from that of selection, these are discussed as separate cases.

8.1 Union, Intersection and Join

The result of comparing the common attributes of a pair of tuples is stored as the comparison value for the pair. A simple qualitative estimate of the comparison value can be obtained in the following way. If all

the common attributes match during the computation of E-join or E-union/E-intersection, the comparison value is *perfect*. If on the other hand, there are any inconsistencies among the common attributes, the value is *probable*.

However, there are applications which require a more precise estimation of the comparison value. We describe a framework for a probabilistic estimation of the comparison value in the remainder of the section.

Assume that a E-join or E-union/E-intersection needs to be performed on two relations, $r(R)$ and $r(S)$. 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, called the *comparison vector*, as $\gamma(t) = \{\gamma_1(t), \gamma_2(t), \ldots, \gamma_n(t)\}$, where tne number of components of $\gamma(t)$ is equal to the number of common attributes between R and S [FS69, Tep68]. (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 for union, intersection and join,

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

8.2 Selection

During selection, a comparison value is assigned to each tuple depending on the selection condition(s) it satisfies. If a tuple fully satisfies the selection conditions, the selection value is *perfect*, otherwise, it is *probable*. A precise way to estimate the comparison value is given below.

Assume E-selection needs to be performed on relation $r(R)$. Let the comparison vector be defined as before except the number of components of $\gamma(t)$ now equals the number of conditions in the selection query. Let there be n such conditions. The result of checking a tuple $r_i \in r(R)$, against the selection query is stored in the comparison vector, $\gamma(r_i)$ which is used to estimate how well the tuple r_i satisfy the selection conditions. Thus, we have,

$$
\begin{aligned}
\gamma_1(r_i) &= 1 \quad \text{if } \textit{condition } 1 \text{ i satisfied} \\
&= 0 \quad \text{otherwise} \\
\gamma_2(r_i) &= 1 \quad \text{if } \textit{condition } 2 \text{ is satisfied} \\
&= 0 \quad \text{otherwise} \\
&\vdots \\
\gamma_n(r_i) &= 1 \quad \text{if } \textit{condition } n \text{ is satisfied} \\
&= 0 \quad \text{otherwise}
\end{aligned}
$$

8.3 Definition of Tuple Probability

The comparison function serves to partition the tuples into classes. Two tuples belong to the same class if their comparison functions are identical. For example, (t_1) belongs to the same class as (t_2) if $\gamma(t_1) = \gamma(t_2)$. The set of all possible realizations of γ constitutes the comparison space, Γ.

The comparison value can now be defined as a function of this vector, that is, $CV(t) = f[\gamma(t)]$. 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 3 *Tuple Probability. Given the results of checking the tuple against the query conditions, the conditional probability that the tuple belongs to $\mathcal{M}$, the result of the query. 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,

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

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

Example 1. (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:

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

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. □

9 The Threshold Probability and Decision Rule

A high value of $p_{tuple}(t)$ means that a large number of common attributes are consistent, in case of union, intersection or join, or the tuple has satisfied a larger number of selection conditions. A low $p_{tuple}(t)$ indicates otherwise. The tuples having a high value of p_{tuple} should be included in the *Result* set and the rest should be excluded. This requires setting a cutoff value, p_{th}, such that all tuples having tuple probabilities greater than this value are considered *good* enough to be included in the *Result*. A correct cutoff value can significantly improve the selectivity of a query, thereby enhancing the power of the E-operators.

Notice that there is a difference between the sets $\mathcal{M}$ and *Result*. The set $\mathcal{M}$ is a hypothetical set that contains the most accurate answer to a query. The set *Result*, is an estimate of $\mathcal{M}$. It may be that *Result* $= \mathcal{M}$, but due to the probabilistic nature of the problem, *Result* might also (erroneously) contain some tuples whose key attributes are not equivalent. This issue is discussed in detail in Section 10.

There are several ways to set a cutoff probability. The users could be requested to provide the threshold value and a decision rule to go with it. For instance, the user could set the threshold at 0.5. A simple instance of rule to go with the threshold could be: "Reject all tuples with p_{tuple} less than the threshold".

An alternate way to calculate the threshold value using cost data is given below [MS86]. According to this decision rule, the tuples with $p_{tuple}(t) = 1$ are to be included in the *Result*, whereas if $p_{tuple}(t) = 0$, the tuples are to be excluded. However, if $0 < p_{tuple}(t) < 1$, then further analysis is required. The decision regarding these tuples are subject to two types of errors.

Type - I. A *type-I* error (or an omission) occurs when $t \in M$ but $t \notin Result$. That is, a tuple that should have been a part of the Result has been accidentally left out.

Type - II. A *type-II* error (or false alarm) occurs when $t \notin M$ but $t \in Result$. This means, an *incorrect* tuple has been included in the *Result*.

Tuples are retrieved in responses to queries. There are costs associated with the inclusion of incorrect tuples and the exclusion of a relevant ones.

Let C_R be the cost of including a wrongly matched tuple in the *Result*. This includes the cost of searching the database, storing of any intermediate and final results as well as any loss the user might incur from using this result. This loss is incurred every time a $type - II$ error is committed. Similarly, let the cost of omitting a relevant tuple from the *Result* be C_O. This is the cost of making a $type - I$ error. Notice that these costs depend on the particulars of the computer, the storage device and also on the possible use of the result of the join.

Thus, the expected cost of including a non relevant tuple in the *Result* is given by $C_R \times (1 - p_{tuple})$. Similarly, the expected cost of omitting a relevant tuple from the *Result* is $C_O \times p_{tuple}$.

Then p_{th} is defined as the probability that minimizes the sum of these costs. This gives:

$$p_{th} = \frac{C_R}{(C_R + C_O)}.$$

This is intuitive because, a higher cost of retrieval, C_R, results in a higher threshold and fewer tuples get included in the Result. As a result, the number of $type - II$ errors is decreased.

A decision rule $d(\gamma)$ can be now defined as a mapping from Γ, the comparison space, to an action space $\{A_m, A_u\}$ where,

$$\begin{aligned} A_m &= \text{include the tuple in the } Result \text{ set} \\ A_u &= \text{do not include the tuple in the } Result \text{ set} \end{aligned}$$

Using the threshold probability obtained above, the optimal decision rule is defined as:

$$d[\gamma(t)] = \begin{cases} A_u & \text{if } 0 \leq p_{tuple}(t) < p_{th} \\ A_m \text{ or } A_u & \text{if } p_{tuple}(t) = p_{th} \\ A_m & \text{if } p_{th} < p_{tuple}(t) < 1 \end{cases}$$

10 An Algorithm to Estimate Tuple Probabilities

The tuple probabilities can be computed most accurately if one has the knowledge of the set $\mathcal{M}$ and the distributions $\Pr\{\gamma(t)\}$ and $\Pr\{t \in \mathcal{M}\}$ and $\Pr\{\gamma(t) \mid t \in \mathcal{M}\}$. Typically, however, these would be some general distributions whose properties may not be available. An iterative algorithm is provided below which will enable us to estimate the probabilities in such situations.

Notice that the expression of tuple probability is recursive. Tuple probabilities are needed to estimate $\mathcal{M}$, but the calculation of the tuple probability assumes the existence of $\mathcal{M}$. The set *Result*, which was introduced in the previous section, is therefore used as the estimate of $\mathcal{M}$. It is important to realize that the contents of the *Result* may vary over time. A tuple that was included in the *Result* because of its common attributes being consistent has to remain consistent to be included in it in future; a tuple that was previously excluded due to inconsistent common attributes may find a place in *Result* when the attributes become consistent. Given the probabilistic nature of the problem, this is intuitively desirable.

In the absence of $\mathcal{M}$, *Result* is utilized. Replacing $\mathcal{M}$ by *Result*, an estimate of p_{tuple} is obtained as follows:

$$p_{tuple}(t) = \Pr\{t \in Result \mid \gamma(t)\}$$

$$= \frac{\Pr[t \in Result,\ \gamma(t)]}{\Pr[\gamma(t)]}$$

$$= \frac{\Pr[\gamma(t) \mid t \in Result].\Pr[t \in Result]}{\Pr[\gamma(t)]}$$

Note that the denominator, that is, the distribution function of the comparison vector is independent of whether *Result* or $\mathcal{M}$ is used. Actually, to begin with, the set *Result* is also unknown and the purpose of the query is to find it. So, the algorithm starts with an initial estimate of *Result*, say, $R^{(0)}$. This estimate can be one of the following:

1. A stored copy of the *Result* from the last computation of the query
2. The outcome of a conventional query
3. Provided by the user

In addition, one needs the distribution function of the comparison vector. Again, the user might provide the information or the system could be made to estimate it. Since the ultimate objective is to free the user as much as possible, it is assumed that the system would maintain the necessary statistics. Option (1) or (2) is assumed to be the initial estimate of *Result*.

The following describes the working of the algorithm. First calculate $\gamma(t)$ for all t. Using them, update $\Pr\{\gamma(t)\}$. Next, using the initial estimate of *Result*, $R^{(0)}$, calculate the expressions $\Pr\{\gamma(t) \mid t \in R^{(0)}\}$ and $\Pr\{t \in R^{(0)}\}$. This enables one to calculate $p_{tuple}(t)$ for all t and using the threshold probability and the decision rule, a new version of the *Result* set, $R^{(1)}$ is obtained. Replacing $R^{(0)}$ by $R^{(1)}$, the above steps are repeated till there are no changes across two iterations. The algorithm is given below in pseudocode.

Step 1. $i := 0$
Step 2. Obtain $\gamma(t)$ for all t.
Step 3. Update $\Pr\{\gamma(t)\}$.
Step 4. Use $R^{(i)}$ to obtain $\Pr\{\gamma(t) \mid t \in R^{(i)}\}$ and $\Pr\{t \in R^{(i)}\}$.
Step 5. Calculate $p_{tuple}(t)\ \forall\ t$.
Step 6. Using $d(\gamma)$ obtain the new *Result* set.
Step 7. If $Result \neq R^{(i)}$
    Then $i := i + 1$
        $R^{(i)} := Result$
        Go to Step 4
    Else Stop

11 Conclusions

The heterogeneous environment will be a prevalent data processing environment for the next decade. Resolution of data heterogeneity problem is central to information retrieval in such an environment. This paper considers the problem of inter-database information retrieval in a heterogeneous setting. The problem arises due to the presence of heterogeneity among data 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, E-union and E-intersection which allows information retrieval across mismatched, incompatible domains. The E-selection allows the retrieval of tuples which partially satisfy the selection conditions. A probabilistic model was presented for estimating the accuracy of the operators in a heterogeneous environment.

We are looking at the following areas for further research:

- Extending the models to cover other sources of data heterogeneity. The extended models will take into account the characteristics of the attributes, viz., their specificity, frequency of updates, number of duplicate copies and correlation with each other.

- Extending the results of this paper to optimize more complicated queries, like combinations of selection and join.
- Device query computation algorithms using graph theoretic results like bipartite graph matching and its variants.
- Perform cost analysis for the proposed algorithms.
- Conduct performance analysis of the algorithms, and implement selective ones, if possible.

References

[AGMS87] Rafael Alonso, Hector Garcia-Molina, and Kenneth Salem. Concurrency control and recovery for global procedures in federated database systems. *IEEE Database Engineering*, 6:129–135, September 1987.

[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.

[BNPS89] E. Bertino, M. Negri, G. Pelagatti, and L. Sbattella. Integration of heterogeneous database applications through an object-oriented interface. *Information Systems*, 14(5):407–420, 1989.

[BOT86] Yuri Breitbart, Peter L. Olson, and Glenn R. Thompson. Database integration in a distributed heterogeneous database system. In *Proceedings of the Second International Conference on Data Engineering, Los Angeles, California*, pages 301–310, February 1986.

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

[BS88] Yuri Breitbart and Avi Silberschatz. Multidatabase update issues. In *Proceedings of ACM SIGMOD International Conference on the Management of Data*, pages 135–142, June 1988.

[CB88] Lawrence T. Cox and Robert F. Boruch. Record linkage, privacy and statistical policy. *Journal of Official Statistics*, 4(1):3–16, 1988.

[CF90] M. Carpenter and M. Fair, editors. *Canadian Epidemiology Research Conference - 1989 Proceedings of the Record Linkage Sessions and Workshop*, Ottawa, Ontario, 1990.

[CH90] J.B. Copas and F.J. Hilton. Record linkage: Statistical models for matching computer records. *Journal of the Royal Statistical Society, Series A*, 153(3):287–312, 1990.

[CS91] Abhirup Chatterjee and Arie Segev. Information retrieval in heterogeneous databases: A probabilistic approach. Technical Report 30754, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, University of California at Berkeley, Berkeley, CA 94720, 1991.

[Dat90] C. J. Date. *Introduction to Database Systems*, volume 1. Addison-Wesley Publishing Company, 5 th edition, 1990.

[DE89] Weimin Du and Ahmed K. Elmagarmid. Quasi serializability: a correctness criterion for global concurrency control in interbase. In *Proceedings of the Fifteenth International Conference on Very Large Databases (VLDB)*, pages 347–355, 1989.

[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.

[Jar89] Matthew A. Jaro. Advances in record linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, June 1989.

[JM90] Mark A. Johnson and Gerald M. Maggiora, editors. *Concepts and Applications of Molecular Similarity*. John Wiley and Sons, Inc., 1990.

[Key79] N. Keyfitz. Information and allocation: Two uses of the 1980 census. *The American Statistician*, 33:45–50, 1979.

[Kim89] Won Kim. Research directions for integrating heterogeneous databases. In *Proceedings of the Workshop on Heterogeneous Databases*. Sponsored by NSF, December 1989.

[LNE89] James A. Larson, Shamkant B. Navathe, and Ramez Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, April 1989.

[Lor83] Maurice Lorr. *Cluster Analysis for Social Scientists*. Jossey-Bass Publishers, 1983.

[LR82] T. Landers and R. Rosenberg. An overview of multibase. In H. Schneider, editor, *Distributed Data Systems*, pages 153–184. North-Holland, 1982.

[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.

[MS86] Haim Mendelson and Aditya N. Saharia. Incomplete information costs and database design. *ACM Transactions on Database Systems*, 11(2):159–185, June 1986.

[MW64] Frederick Mosteller and David L. Wallace. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley Publishing Company, Inc, 1964.

[Pu87] Calton Pu. Superdatabases: Transactions across database boundaries. *IEEE Database Engineering*, 6:143–149, September 1987.

[Pu88] Calton Pu. Superdatabases for composition of heterogeneous databases. In *Proceedings of the Fourth International Conference on Data Engineering, Los Angeles, California*, pages 548–555, February 1988.

[Pu91] Calton Pu. Key equivalence in heterogeneous databases. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan*, pages 314–316. Sponsored by IEEE Computer Society and The Information Processing Society of Japan, April 1991.

[Ros89] Paul R. Rosenbaum. Optimal matching for observational studies. *Journal of the American Statistical Association*, 84(408):1024–1032, December 1989.

[RS90] F. Rabitti and P. Savino. 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.

[SGJ86] Richard J. Smith, Rebecca Z. German, and William L. Jungers. Variability of biological similarity criteria. *Journal of Theoretical Biology*, 118(3):287–293, February 1986.

[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.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

[SSG+91] Ashoka Savasere, Amit P. Sheth, Sunit K. Gala, Shamkant B. Navathe, and Howard Marcus. On applying classification to schema integration. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan*, pages 258–261. Sponsored by IEEE Computer Society and The Information Processing Society of Japan, April 1991.

[Taf70] R.L. Taft. Name search techniques. Technical report, New York State Identification and Intelligence System, New York, 1970.

[Tep68] B. J. Tepping. A model for optimal linkage of records. *Journal of the American Statistical Association*, 63:1321–1332, December 1968.

[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.

END

DATE
FILMED
9/02/92