Using Field Specifications to Determine Attribute Equivalence in Heterogeneous Databases

Wen-Syan Li
acura@eecs.nwu.edu

Chris Clifton
clifton@eecs.nwu.edu

Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois, 60208-3118

Abstract

One step in integrating heterogeneous database systems is matching equivalent attributes: Determining which fields in the two databases refer to the same data. We see three (complementary) techniques to automate this process: Synonym dictionaries that compare field *names*, design criteria that compare field *specifications*, and comparison of data *values*. In this paper we present a technique for using field specifications to compare attributes, and evaluate this technique on a variety of databases.

1 Introduction

One problem in developing a global schema for heterogeneous databases is to determine which fields are equivalent between the databases. Attributes are compared in a pairwise fashion to determine their equivalence. Manually comparing all possible pairs of attributes is an unreasonably large task, especially since most pairs do *not* represent the same information. Simple ad-hoc guesswork, on the other hand, is likely to miss some attributes that could map to the same global attribute.

Other problems encountered here are that synonyms occur when objects with different names represent the same concepts, and homonyms occur when the names are the same but different concepts are represented.

Several approaches have been proposed to determine attribute equivalence:

Searching a synonym lexicon: It is to compare objects in a pairwise fashion by consulting a lexicon of synonyms. Systems have been developed to automate database integration. One that has addressed the problem of attribute equivalence is MUVIS (Multi-User View Integration System) [HR90]. MUVIS determines the degree of similarity and dissimilarity of two objects during a *pre-integration* phase. Object equivalence is determined by comparing the aspects of each (such as class names, and member names) and computing a weighted probability of similarity and dissimilarity. This approach works well for homonyms. However, different objects can have different synonyms that are not easily detected by inspection.

Comparing attribute values and domains: [SLCN88, LNE89] discussed how relationships and entity sets can be integrated primarily based on their domain relationships: EQUAL, CONTAINS, OVERLAP, CONTAINED-IN, and DISJOINT. However, determining such relationships can be time consuming and tedious [SL90]. Other problems with their approach is the ability to handle faults and the fact that data may not always reflect real attribute relationships since domain relationships change over time.

A technique for determining the degree of similarity using field specifications is presented in Section 2. In Section 3 we give experimental results from running this algorithm on two pairs of real databases.

2 Technique for Determining a Degree of Similarity and Dissimilarity

Given a database design application, different designers should tend to have similar schema and constraint design because they should have the same technology and knowledge about designing a "good" database. Thus information about attributes; such as length, data types, and constraints; can be used as indicators to determine the likelihood that two fields are equivalent. We categorize the characteristics of a database into two groups:

0-8186-3710-2/93 $3.00 © 1993 IEEE

- Database schema specifications, which include data types, length, key fields, and "supplemental data types" such as format specifications (common examples are EDTWRD and EDTCDE specifications [1]).
- Data constraints, which include primary keys, foreign keys, candidate keys, value and range constraints, disallowing null values, access restrictions, etc.

The algorithm for determining a degree of similarity and dissimilarity is:

Step 1: $Degree\ of\ similarity = Sim_{field\_specification} = 0$
$Degree\ of\ dissimilarity = Dis_{field\_specification} = 0$

Step 2: Compare two fields characteristics. If they match (or don't match), assign the appropriate probability value from Table 1 to $Sim_{field\_specification}X$ or $Dis_{field\_specification}X$.

Step 3: $Sim_{field\_specification} = Sim_{field\_specification} + (1 - Sim_{field\_specification}) * Sim_{field\_specification}X$
$Dis_{field\_specification} = Dis_{field\_specification} + (1 - Dis_{field\_specification}) * Dis_{field\_specification}X$
Repeat step 2 and step 3 on all applicable comparisons.

Step 4: These numbers are then normalized based on the field specification information available from the particular DBMSs.

Normalization is necessary here because the numbers of rules applied may vary. They depend on the availability of schema design information.

For example, if only 7 of the 10 characteristics listed in Table 1 are used, 7 rules can be applied to determine similarity and 5 rules can be applied to determine dissimilarity. The maximum possible degree of similarity and dissimilarity are 0.36 and 0.55 [2] respectively. If two objects have a normalized degree of similarity greater than 0.8 [3] (the similarity is greater than 80% of the maximum possible degree of similarity; 0.29 in this example) and a normalized dissimilarity is less than 0.2 (less than 20% of the maximum possible degree of dissimilarity; 0.11 in this example), then the objects are presumed likely to be equivalent. If normalized similarity is less than 0.8 and normalized dissimilarity is greater than 0.2, they are presumed non-equivalent. Otherwise we draw no conclusion.

[1]EDTWRD (edit word) is used to specify a format for a particular field. EDTCDE (edit code) provides specific formats for numeric fields.

[2]The maximum possible *Sim* and *Dis* are approximately 1 when all the 10 characteristics are used.

[3]These numbers come from MUVIS [HR90]. The similarity value using this technique can be combined with similarity values based on attribute names using a synonym lexicon. Assuming that values are independent, the following can be used to obtain overall similarity: $Similarity = Sim_{field\_name} + (1 - Sim_{field\_name}) * Sim_{field\_specification}$.

Rules	*Sim*	*Dis*
Schema Specifications		
Similar field length	+0.18 [4]	
Dissimilar field length		+0.2
Same data type	+0.12	
Different data types		+0.2
Both are key fields	+0.06	
Both are unique keys	+0.12	
Only one is an unique key		+0.08
Only one is a key field		+0.08
Neither are key fields	+0.03	
Neither have EDTCDE	+0.03	
Have different EDTCDE		+0.12
Only one has EDTCDE		+0.2
Both have same EDTCDE	+0.12	
Neither have EDTWRD	+0.03	
Have different EDTWRD		+0.12
Only one has EDTWRD		+0.2
Both have same EDTWRD	+0.12	
Constraints		
Both fields refer to another relation	+0.12	
Only one field refers to another relation		+0.16
Neither refer to another relation	+0.03	
Similar value constraints	+0.09	
Both have value constraints	+0.06	
Different value constraints		+0.08
Only one field has a value constraint		+0.16
Neither have value constraints	+0.03	
Both have range constraints	+0.06	
Only one field has a range constraint		+0.16
Neither have range constraints	+0.03	
Both have no null values constraint	+0.12	
Neither have no null values constraint	+0.03	
Only one has no null values constraint		+0.16
Both have public R/W prohibited	+0.12	
Both have public Write prohibited	+0.09	
Neither have file access constraints	+0.03	
Only one has file access constraints		+0.16
With different file access constraints		+0.12

Table 1: Probability values of degree of similarity and dissimilarity for schema specification and constraint comparisons

3 Experimental Results

We tested this technique using two pairs of existing databases [5]. The initial results do show applicability of using field specifications as indicators for determining field equivalence.

3.1 AS/400 databases

The first pair of sample databases are two field reference files runing on IBM AS/400. The characteristics

[4]These probability values are statistics from experiments on real databases.

[5]The results from running experiments on heterogeneous databases will be more convincing. However, we have not been able to do so due to the availability of existing databases.

available for us to use are data types, length, range and value constraints, and EDTCDE and EDTWRD specifications [6].

There are 10 fields in the database that record general marketing activity information and 41 fields in the database that record telemarketing activity information respectively. Thus, there are 410 pairs of fields being compared (Eight of these pairs were equivalent.) The algorithm recognized 62.50% of equivalent pairs with a fault rate of 89.13% (41 pairs of non-equivalent pairs were consiered equivalent) and 9 pairs (2.19%) could not be determined. The high fault-rate is because there are only four characteristics available to determine attribute equivalence. The algorithm can not differentiate those attributes whose data types are Date or Time, which are equivalent in length and data type.

The algorithm was largely successful in eliminating non-equivalent pairs. It eliminated 88.31% of non-equivalent pairs with a fault-rate of 0%.

3.1.1 Sybase databases

The second pair of databases are Sybase Databases running on Sun-4 SPARC stations.

The available characteristics of these databases include data types, length, key fields, value and range constraints, file access restrictions, foreign key constraints, and not null value constraints. There are 7 fields in the database that record project document information and 6 fields in the database record project meeting information. Thus, there are 42 pairs of fields being compared, of which two pairs are equivalent.

Only one non-equivalent pair was considered to be *likely to be equivalent*. This is as we would expect, because a wider variety of characteristics are available so that the algorithm can differ non-equivalent pairs from equivalent pairs more effectively. Again the algorithm was effective in eliminating a substantial number of non-equivalent pairs. It eliminated 62.50% of non-equivalent pairs with a fault-rate of 0%. The undetermined pair rate was 35.71%.

3.2 Applicability

Adjusting the similarity and dissimilarity cutoff points made some difference with these two pairs of databases. More experiments are needed to determine optimal cutoff points, however histograms for degree of similarity show a high correlation between

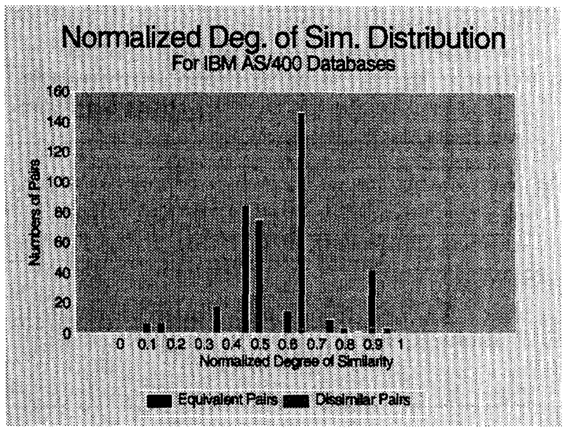

Figure 1: Similarity Distribution of AS/400 Databases.

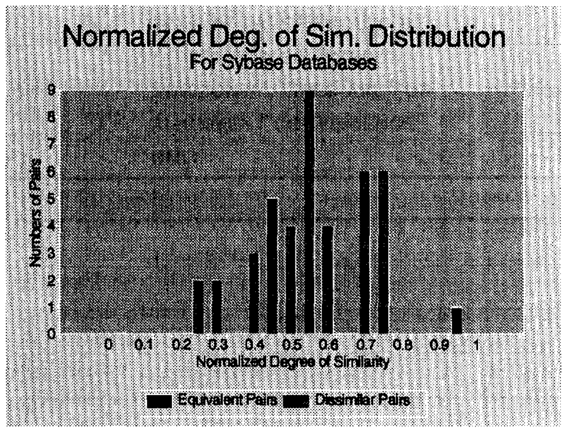

Figure 2: Similarity Distribution of SyBase Databases.

similarity and equivalence (Figures 1 and 2). Dissimilarity is an even more effective discriminator for these experiments. The histograms in Figures 3 and 4 show that the dissimilarity of non-equivalent pairs was higher than that of equivalent pairs. We were quite successful in eliminating dissimilar pairs. 88% of the non-equivalent pairs for the AS/400 databases were judged non-equivalent, as were 62% of non-equivalent pairs for the Sybase databases. The fault rates on non-equivalent pairs were 0% for both databases. Therefore, at least for this example, well over half of all pairs could be eliminated from consideration using this method alone. At the very least, this allows more computationally expensive pre-integration methods [LNE89, SLCN88, She88, SG89, SL90] to work on a smaller problem.

[6]The IBM AS/400 databases have all the design information described in Section 2 available on the system. However, some of this information was not available to us.

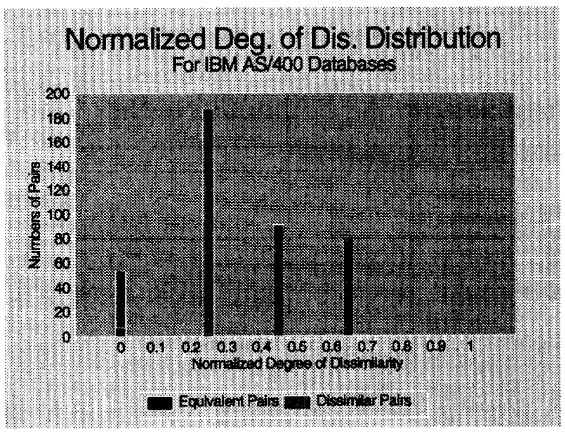

Figure 3: Dissimilarity Distribution of AS/400 Databases.

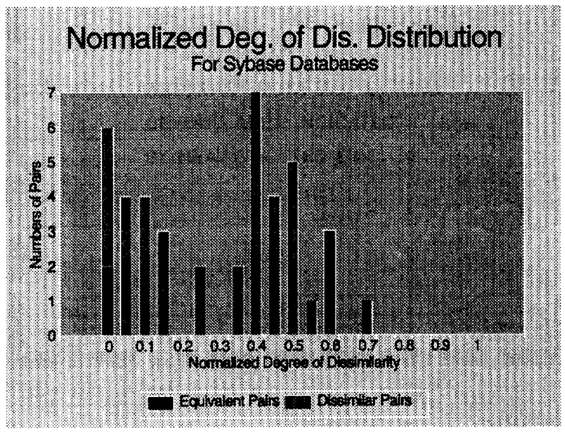

Figure 4: Dissimilarity Distribution of SyBase Databases.

4 Conclusions

We show a technique that extends the method of MUVIS by utilizing field specification characteristics of attributes. The experimental results on two pairs of existing databases show the applicability of our technique. The advantages of our approach (over simply comparing field names or using attribute domain relationships) include: the effect of synonyms and homonyms will be different (the problem is with *structural* synonyms/homonyms, rather than *dictionary* synonyms and homonyms.) and it solves the time consuming problem of determining the attribute (domain) relations in [LNE89, SLCN88, She88, SG89, SL90] by eliminating most of the non-equivalent attribute pairs based simply on field specification level information.

Acknowledgements

We would like to thank Serge Ioffe of Marketing Information Systems in Chicago for providing the AS/400 database field reference files, and Marke Ellis and Zhaoying Lin of Reuters Information Management Systems in Oak Brook, Illinois for providing the Sybase Database files that are used in this paper.

References

[HR90] Stephen Hayne and Sudha Ram. Multi-user view integration system (MUVIS): An expert system for view integration. In *Proceedings in the 6th International Conference on Data Engineering*, pages 402–409. IEEE, February 1990.

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

[SG89] Amit Sheth and Sunit K. Gala. Attribute relationships: An impediment in automating schema integration. In *Processings of NSF Workshop on Heterogeneous Database Systems*, Evanston, IL, December 1989.

[She88] Amit Sheth. Managing and integrating unstructured data problem of representation, features, and abstraction. In *Processings of 4th International Conference on Data Engineering*, pages 598–599, Los Angeles, CA, February 1988.

[SL90] Amit Sheth and James Larson. Federated database systems for managing distributed heterogeneous, and autonomous databases. *Computer Surveys*, 22(3):183–236, September 1990.

[SLCN88] Amit Sheth, James Larson, A. Cornelio, and S. B. Navathe. A tool for integraing conceptual schemas and user views. In *Processings of 4th Inernational Conference on Data Engineering*, Los Angeles, CA, February 1988.