A SEMANTIC DATA DICTIONARY METHOD FOR DATABASE SCHEMA INTEGRATION IN CIESIN

N. Hinds, Y. Huang & C. Ravishankar
University of Michigan, Ann Arbor, Michigan 48109-2122

Abstract

CIESIN (Consortium for International Earth Science Information Network) is funded by NASA to investigate the technology necessary to integrate and facilitate the interdisciplinary use of Global Change information. A clear part of this mission includes providing a link between the various global change data sets, in particular the physical sciences and the human (social) sciences.

The typical scientist using the CIESIN system will want to know how phenomena in an outside field affects his/her work. For example, a medical researcher might ask: how does air-quality effect emphysema? This and many similar questions will require sophisticated semantic data integration. The researcher who raised the question may be familiar with medical data sets containing emphysema occurrences. But this same investigator may know little, if anything, about the existence or location of air-quality data. It is easy to envision a system which would allow that investigator to locate and perform a "join" on two data sets, one containing emphysema cases and the other containing air-quality levels. No such system exists today.

One major obstacle to providing such a system will be overcoming the heterogeneity which falls into two broad categories. "Database system" heterogeneity involves differences in data models and packages. "Data semantic" heterogeneity involves differences in terminology between disciplines which translates into data semantic issues, and varying levels of data refinement, from raw to summary.

Our work investigates a global data dictionary mechanism to facilitate a merged data service. Specifically, we propose using a semantic tree during schema definition to aid in locating and integrating heterogeneous databases.

1 INTRODUCTION

CIESIN (Consortium for International Earth Science Information Network) is participating in a Federally funded project to explore technology necessary to integrate and facilitate the use of global change information. The work includes developing mechanisms to make this information available to scientists, policy-makers, and other user communities in their roles of researching and managing global change.[11]

CIESIN will serve as a link between relevant databases and an international user community in facilitating access, distribution and use of derived scientific information in the pursuit of understanding and predicting global environmental change.[11]

The general challenge CIESIN faces is to develop information management systems which support context-based information retrieval, heterogeneous and distributed databases, and "seamless" user interfaces. Ideally, it must allow users to browse and search through information domains using sophisticated querying techniques that will include imprecise queries, user-directed query processing, and queries that use similarity measures in order to retrieve data.

The CIESIN vision mandates a non-predatory approach to data acquisition, so the CIESIN domain will be a collection of autonomous databases. These databases will differ considerably from one another not just in their contents, but also in their characteristics. For example, they will all have different schemas, semantic properties, and attributes. That is, CIESIN will administer a "heterogeneous database" domain.

Our work investigates data dictionary extensions to facilitate a merged data service within this complex heterogeneous environment. Specifically, we present a semantic schema definition technique used to locate and integrate heterogeneous databases. The next section provides a more detailed problem definition and examples. In section 3 we explore some state-of-the-art heterogeneous database solutions. Section 4 describes our techniques for semantic schema description in the CIESIN environment. Finally, in section 5 we conclude with the current status and our proposed future work.

2 CIESIN EXAMPLE

In this section we will describe an expected scientist interaction with the CIESIN information system. This example is then used to illustrate some of the technical issues faced by the CIESIN system developers, and some potential solutions.

In our scenario, a medical scientist is interested in studying how changes in particulate pollution levels may affect the number cases of emphysema in the city of Metropolis over the past few years. The information about emphysema cases is available in a epidemiology database (data set), while the measurements of particulate pollution levels are available in an air-quality database. Since much of the current scientific data is stored in tabular formats which fit the relational model quite well, we have chosen the relational model as our platform for this preliminary work.

The problem of semantic heterogeneity arises within CIESIN because existing data sets can not be used to answer multidisciplinary queries directly. In our scenario, Tables 1 and 2 represent the data sets retrieved from two data sources (the epidemiology data set and the air-quality data set).
Some typical queries that the scientist may wish to make are as follows:

Query1: What is the correlation between the measurements of particulate pollution levels and those of emphysema in the city of Metropolis?

Query2: What is the change of the number of emphysema cases between 1988 and 1989 in the city of Metropolis?

Query3: What are the changes of particulate pollution levels and the number of emphysema cases between 1988 and 1989 in the city of Metropolis?

Year	Emphysema Cases
1988	1,052
1989	1,503
1990	2,162

Table 1: Data Set (1) from Epidemiology Database

Date	Particulate Concentration (gm/liter)
06/01/88	0.00210
12/01/88	0.00220
06/01/89	0.00222
12/01/89	0.00228
06/01/90	0.00232
12/01/90	0.00240

Table 2: Data Set (2) from Air-Quality Database

The epidemiology data set holds the total number of emphysema cases in the city each year. The air-quality data set provides measurements of the particulate concentrations every six months. However, these two data sets do not provide direct answers to any of the queries mentioned above. To answer the above queries, we must perform a "semantic join" on these two data sets. A semantic join is similar to the familiar relational theory join operation. Where the regular join performs field value matching to combine records, the semantic join must perform field meaning matching. For example, Table 3 shows the two data sets semantically joined on the meaning of the "Date" and "Year" fields. Table 3 can also be considered a "view", which merges the data in Tables 1 and 2.

This is a simple example of data integration across different domains, but it il-

Year	Particulate Conc.	Emphysema Cases
1988	0.00215	1,052
1989	0.00225	1,503
1990	0.00236	2,162

Table 3: The Integrated Data Set For Query 1

From	To	Increase in Emphysema Cases
1988	1989	451
1989	1990	656

Table 4: The Integrated Data Set For Query 2

lustrates a most important aspect of the CIESIN mission. Now, data retrieved from different sources may have different data representations, different data units, and different data semantics. A first step in data integration is data conversion. In our example, the data representation corresponding to the attribute "Year" in the epidemiology data set are very different from that corresponding to the attribute "Date" in the air-quality data set.

The data corresponding to the attribute "Year" may be represented as integers (e.g. 1988), while the data corresponding to the attribute "Date" may be represented as strings (e.g. "08/01/88"). The data units of these two attributes are also different (year vs. month/day/year). The data semantics of the two attributes are also different. The attribute "Year" indicates which year the data for the number of emphysema cases belongs to, while the attribute "Date" shows the date on which air quality was measured.

Data integration must be accomplished by comparing the value corresponding to the "Year" attribute and the value corresponding to the "Date" attribute. However, such data comparison is meaningful only if two data items are on a common referential basis, i.e., their data representations, data units and data semantics are the same. Therefore, data of the "Date" attribute must be converted to the "Year" attribute before two data sets can be integrated. But, this may also involve "data refinement". For example, two half-year air-quality measurements may first be averaged to obtain a yearly air-quality value, then the two data sets merged. This is a form of data refinement which usually requires considerable domain knowledge, and it may be necessary to involve human experts in the process.

The data integration problem is not limited to the difficulties of conversion among the different data representations or data units. It may also involve data refinement. For example, to answer Query2, we must compute the change in the number of emphysema cases (table 4).

To answer Query3, we must perform all the data conversions and data refinements mentioned above on both data sets before they can be properly integrated. The air-quality data set must be converted to a yearly basis, and the change of particulate concentration must be computed. Then, we can integrate this result with the result from Query2 to answer Query3. Table 5 contains the data integration results for Query3. From the results of Query3, the user can clearly see the correlation between the increase of particulate concentration and the the increase of emphysema cases.

These examples present simple illustrations of some problems CIESIN must overcome in order to provide data integration service. In the next section we will review current work in heterogeneous database systems and its limitations. In section 4, we

From	To	Change in Emphysema Cases	Change in Particulate Conc.
1988	1989	451	0.00010
1989	1990	656	0.00011

Table 5: The Integrated Data Set For Query 3

will present our approach to CIESIN's semantic integration problem.

3 PREVIOUS WORK

Sheth[13] divides database heterogeneity into data model and semantic differences, and operating systems and hardware differences. Operating systems and hardware differences are important and have been addressed with standards or de facto standards, including the ISO Remote Data Access (RDA) standard,[1] and IBM's CICS.[9] Data model differences include differences in query language and structures. These issues have also been extensively studied. Our primary focus is on semantic heterogeneity.

Semantic heterogeneity arises from differences in the meaning of related data. Consider the example given by Sheth.[13] Two independent databases, DB1 and DB2, contain meal cost data. DB1 has data for restaurants and DB2 has data for boarding houses. The attribute MEAL_COST of relation RESTAURANT in database DB1 gives the average cost of a meal per person in a restaurant "without" service charge and taxes. The same attribute name in relation the BOARDING in database DB2 expresses average cost of a meal per person "including" service charge and taxes. Semantic heterogeneity makes comparing or joining the two fields very tricky.

To overcome this problem, most state-of-the-art systems use what Sheth[13] calls a "transforming processor" in conjunction with a global schema or virtual view. Figure 1 shows the relationship between the transformation processor and the schemas. MULTIBASE[8] is an early example of this architecture (see Figure 2). MULTIBASE divides data heterogeneity into semantic and database integration issues, and the "homogenization" of local databases. MULTIBASE uses the DAPLEX[14] functional data modeling and query language to describe (homogenize) local schemas, i.e., present the global system with a homogeneous relational view of the local data. DAPLEX flexibility makes it possible to isolate differences in data model and access methods in the local schema.

When a user performs a DAPLEX query on the global schema, the global data manager (GDM) transforms and distributes the query to the local database interfaces (LDI). Using the local DAPLEX schema the LDI transforms the query into the language of the local database system. Schema integration is performed by the GDM Transformer with the aid of "view derivations". View derivations are a set of rules (derivation operators) expressed as a DAPLEX query. Derivation operators can also be called mapping functions or operators. The global schema or view is a logical connection between, and

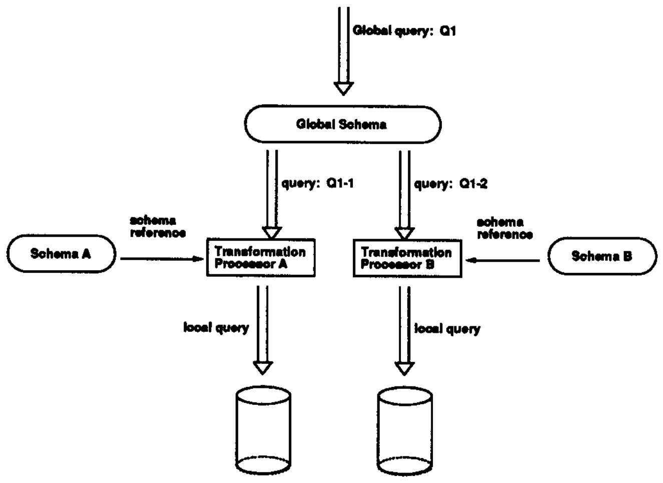


Figure 1: Conventional Heterogeneous Schema Integration

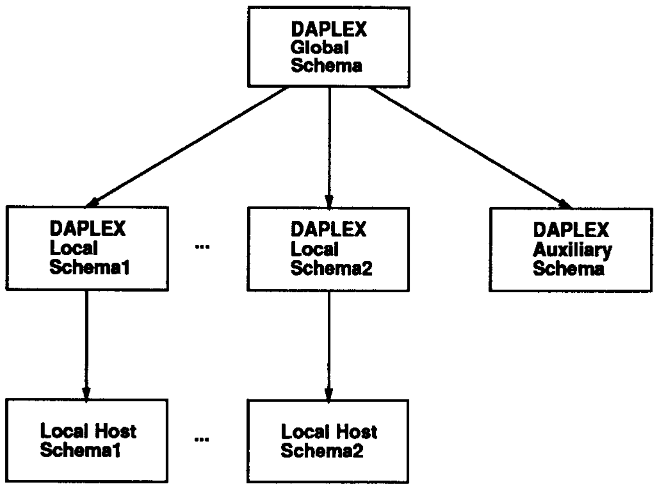


Figure 2: MULTIBASE Schema Architecture

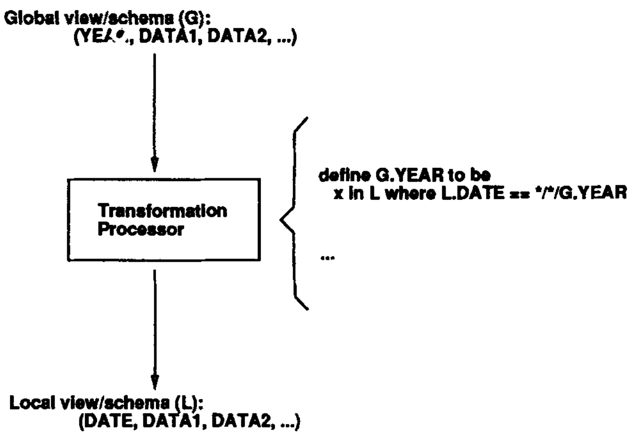


Figure 3: Sample Global Schema Transformation Rule

presentation of, any number of the underlying local schemas. View derivations are used to define views in terms of local schemas. MULTIBASE also incorporated auxiliary databases to store information necessary to perform semantic mappings. The resent Pegasus[2] system work is comparable to MULTIBASE .

Figure 3 uses a DAPLEX-like view description language to illustrate this process with the fields "YEAR" (e.g., 1876, 1991) in a global schema and "DATE" (e.g., 4/23/91) in a local schema. This example of data refinement requires a global schema designer (with "intelligence") to reconcile the semantic differences and describe a mapping from values in the global schema to those in the various private or foreign schemas. A number of other systems are based on this mapping technique.[3,4]

Heimbigner and Mcleod[6] describe a federation architecture with no global schema. The federation is organized as equal, autonomous databases or components, with sharing controlled by explicit interfaces. Each component/site is able to, in effect, build its own "global" schema that is best suited to its needs.

Each federation has a single "federation dictionary" (FD) which maintains data about the member components. The FD:

- supports entry and removal of components in the federation
- has no direct control over the components
- does not mediate communication with components

Components define "private schemas" which describe data stored in the component. These schemas most resemble traditional non-federated database schemas. An "export schema" is then defined for the data sets to be made available to the rest of the federation. Federation participants search the FD to find schemas of interest and then import their exported schemas. Data within the exported schemas is then combined with view mapping operations (similar to those in MULTIBASE) to construct an "import schema" (or view).

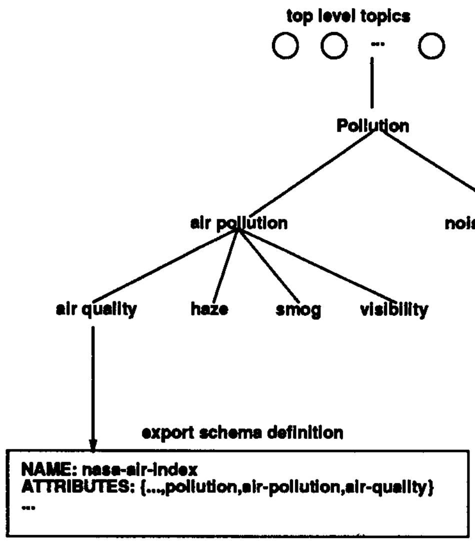


Figure 4: Semantic Tree Structure

4 SEMANTIC INTEGRATION

CIESIN's data semantics problems are unique. The diversity in its scientific data makes a single global schema impractical, if not impossible within the CIESIN environment. Further, any CIESIN solution must accommodate new data sets, and modifying a centrally administered global schema would make this difficult. Another important difference is the nature of the CIESIN user. The scientist using this system will have knowledge of a particular field of study, but limited knowledge outside that field. So, CIESIN must provide "knowledge discovery" services.

In some sense, the CIESIN system will be a data dictionary (DD) of schemas, similar to a FDD. Users will access the DD in search of data sets. For example, a researcher might want to locate all data sets having information on "emphysema". Differences in variable names and a general lack of descriptive information might make this search difficult in heterogeneous database systems described in the last section. The CIESIN system will use semantic information to facilitate discovery.

To address these problems, we propose a global data dictionary (GDD) for CIESIN. We will start with the basic GDD presented in the last section. The export schemas in the GDD will be extended to also contain a set of keyword attributes providing semantic annotation. This will be used to providing discovery mechanisms for exported schemas. The export schema will also contain "type tree" definitions of the schema fields to be used to partially automate the view creation process.

In the next two sections we describe these extensions to the basic GDD.

4.1 Semantic Hierarchy

Each data set administrator in the CIESIN federation will have his or her own interpretation of the data set contents. To allow discovery we had to provide a common framework in which users could describe data sets. Annotating the schema definitions with descriptive (semantic) attributes imposes a level of standardization required for any shared semantic understanding. A set of unique keywords or semantic attributes will used. Keywords will be stored as an unordered set of values with the schema definition in the GDD. The user will have the option of specifying attribute-based searches or navigating through a hierarchy (Semantic Tree) of attributes to locate the desired data sets.

The Semantic Tree (ST) is a "classification system". Classification systems describe groupings of similar objects and show relationships between dissimilar groups.[10] A simple classification structure looks like a hierarchy, but more complex relationships can generate network structures. "Hierarchical classification" relationships are based on the principles of subordination (specialization) and inclusion. The Dewey decimal system is a typical hierarchical classification system.

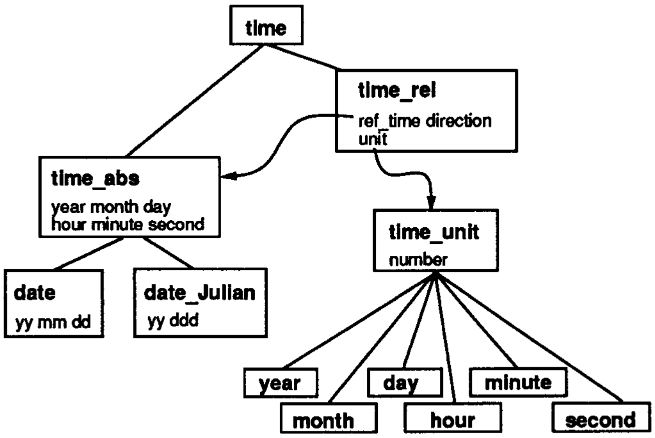
The system we propose is a hierarchical classification, where the universe of objects is divided into successively narrower classes. The interior nodes represent unique semantic keywords (classes), e.g., "epidemiology", "emphysema". The children of any node subdivide the class of the parent. The leaves of the ST contain directory information for the actual data set schemas. When a data set administrator is defining a schema to be included in the CIESIN system, that person will navigate the ST to annotate the schema with semantic attributes. The effect of this navigation process is to place a data set schema at a leaf of the ST and create a database of schemas (a data dictionary). The semantic keywords in the navigation path become part of the schema definition and will aid users who subsequently search the CIESIN system looking for data sets on a certain topic (data sets with certain semantic attributes).

Figure 4 shows a partial semantic tree representation including a class for our air-quality data set. Clearly, the key to this solution is proposing a sufficiently complete set of keywords and structure in the ST. A draft list and hierarchy of keywords is being developed by CIESIN.[5]

4.2 Type Tree

Since CIESIN has no global schema, mapping functions can not be implemented in advance. Schemas must be self-describing (via semantic attributes), so mappings can be constructed at query or view-creation time. The type tree (TT) provides the schema creator with object hierarchies of "field types" (type objects) to be used during the schema definition. Each type object in a TT not only contains semantic attributes, but a form of view mapping operators as well. The operators are used for constructing new type objects in terms of old ones. We are working on a mapping language which will automate a large portion of the transformation processor synthesis.

Semantic attributes in an object are either inherited from its parent object or created for specializing the object. For a user-defined attribute, the schema creator must provide two attribute-mapping functions which specify how the user-defined attribute can be mapped to/from some system attributes. If these mapping functions are not provided, the system will not be able to perform data conversion related to this user-defined attribute.



```
date.yy  = { date.year % 100 }
date.mm = { date.month }
date.dd  = { date.day }

date_Julian.yy = { date.yy }
date_Julian.ddd = {
   for (tmp=0, i = 1; i <= date_Julian.month; i++)
        tmp = tmp + days_in_month[i];
   date_Julian.ddd = tmp + date_Julian.day;
}

date_Julian.month = {
    i = 0; tmp = date_Julian.day
    do {
      tmp = tmp - days_in_month[i];
      i++;
    } (tmp <= 0)
    date_Julian.month = i;
}
```

Figure 5: Type Tree Example

Figure 5 illustrates the TT for "time". The "time" object in the TT represents the generic concept of time. The time specified in the TT can be either "absolute" or "relative". Thus, two system objects, "time_abs" (absolute time) and " time_rel" (relative time), are defined to reflect this model. The absolute time represents a time instance relative to 00:00:00, January 1, 0 A.D.. The relative time is a time instance relative to some absolute time. Users can define new time objects with respect to system objects or existing user-defined objects. In our example, the objects "date" and "date_Julian" are user-defined objects. The mapping functions for these two objects are also provided.

We will use a C language subset to describe attribute mapping functions. We require that if a schema creator defines a mapping function from type X to type Y, then that person also defines a mapping ("complement") function from type Y to type X. While we require that mappings be two-way, we do not require that the two directions of the mapping be strict mathematical inverses of each other. We would prefer to work with such inverse mappings, but we recognize that there may be cases where it is impossible to devise strict inverses.

The data conversion from one attribute to the other can be realized by a sequence of attribute mapping operators/functions. This sequence of operators/functions is called the "conversion path". By requiring the user to define complement functions, we can always construct a reverse conversion path by replacing all the operators with their complements and applying them in reverse order. Therefore, the data conversion process in TT level is always reversible. Examples of reversible operators are '+', '-', '*', '/', and '='. The reversibility of operators is important because it enables our system to construct the conversion functions automatically. The fact that any user-defined attribute can be converted to/from some system attributes, guarantees the existence of some conversion function to perform conversion between any two attributes with compatible units. The conversion function can be constructed by chaining all the operators and mapping functions along the conversion path.

The units of time are defined as a type tree ("time_unit"). Each unit object in the "time_unit" subtree contains at least one conversion function to some other unit object. To support conversion between any two unit objects, all the objects in the subtree must be "strongly connected" by some conversion path. Some unit object also contains the constants that are necessary for the data conversion. For example, the "month" object contains an array of constants to store the number of days in a month ("days_in_month"), which is shared by all its derived (children) objects.

The conversion at the TT level is restricted to data without data refinement (see section 2). The nature of the data refinement process is different from the data conversion process. Data refinement functions/operators are similar to the view mapping or derivation operators discussed in section 3. The data refinement function may not be reversible (from high-resolution data to low-resolution data). Because it is impossible for a system to predict a user's intention for the data refinement process, we rely on the user to select the proper refinement functions at view creation time. Given the refinement function, the system will perform the data refinement incorporated with the necessary data conversion. Some typical refinement functions are average functions and summation functions.

Together, the attribute mapping and refinement operators define the schema view definition language. It is the aim of this work to automate the processes of field mapping or transformation processor construction. We define as much semantics as possible in the TT. But, domain specific knowledge makes it impossible to create views without user intervention at creation time.

5 STATUS & FUTURE WORK

This paper describes some of the information systems work being performed within the CIESIN project. Research in heterogeneous database semantics is now increasing and many of the issues discussed in this paper remain unsolved. Although completely automated meaningful semantic join is difficult (if not impossible) without user intervention, the future of heterogeneous databases is in methods and tools to aid semantic intergation.[7]

Future work falls into four categories: refinements to the ST , implementation of the schema specification language, implementation of the user query or view language, and incorporating this work into a complete distributed system. Some of these future directions are outlined below.

Because the ST and TT are based on a strict hierarchy, they have many limitations. As the CIESIN keyword list expands cross referencing is a natural extension. Multiple inheritance techniques which solve many of these problems have been discussed by Sciore.[12] There is also a general question whether we will be able to define a sufficiently complete class hierarchy for the ST or TT.

There are many details of the attribute mapping and refinement functions to be considered. The language should perform "fuzzy operations" which would allow ranges of values to satisfy a query.

Finally, when we implement the semantic and type trees, we plan to do so on some existing systems such as DAPLEX or Pegasus.

References

[1] Remote data access protocol. ISO/TC 97/SC 21/WG 3, International Standards Organization, 1987.

[2] R. Ahmed et al. The Pegasus heterogeneous multibase system. *IEEE Computer*, 24(12):19–27, December 1991.

[3] S. Cammarata. An intelligent information dictionary for semantic manipulation of relational databases. In J. Schmidt, S. Ceri, and M. Missikoff, editors, *International Conference on Extending Database Technology*, pages 214–230, March 1988.

[4] C. Chung. DATAPLEX: An access to heterogeneous distributed databases. *Communications of the ACM*, 33(1):70–80, January 1990.

[5] CIESIN global environmental directory: Phase 1 concept demonstration design document. Technical report, CIESIN, September 1991.

[6] D. Heimbigner and D. Mcleod. A federated architecture for information management. *ACM Transactions on Information Systems*, 3(3), July 1985.

[7] W. Kim and J. Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, 24(12):12–18, December 1991.

[8] T. Landers and R. Rosenberg. An overview of MULTIBASE. In H. Schneider, editor, *Distributed Data Bases*, pages 153–183. North-Holland, 1982.

[9] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[10] R. Prieto-Diaz and P. Freeman. Classifying software for reusability. *IEEE Software*, 4(1), January 1987.

[11] N. Roller et al. Draft: CIESIN mission statement. Technical report, CIESIN, June 1991.

[12] E. Sciore. Object specialization. *ACM Transactions on Information Systems*, 7(2):103–122, April 1989.

[13] A. Sheth and J. Larson. Federated database systems for managing distributed heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[14] D. Shipman. The functional data model and the data language DAPLEX. *ACM Transactions on Database Systems*, 6(1):140–173, March 1981.