Journal of Biomedical Informatics **34,** 285–298 (2001)
doi:10.1006/jbin.2001.1024, available online at http://www.idealibrary.com on IDEAL®

METHODOLOGICAL REVIEW

Heterogeneous Database Integration in Biomedicine

Walter Sujansky

ePocrates, Inc., 1927 Eaton Avenue, San Carlos, California 94070

E-mail: wsujansky@epocrates.com

Received June 8, 2001; published online January 22, 2002

The rapid expansion of biomedical knowledge, reduction in computing costs, and spread of internet access have created an ocean of electronic data. The decentralized nature of our scientific community and healthcare system, however, has resulted in a patchwork of diverse, or heterogeneous, database implementations, making access to and aggregation of data across databases very difficult. The database heterogeneity problem applies equally to clinical data describing individual patients and biological data characterizing our genome. Specifically, databases are highly heterogeneous with respect to the data models they employ, the data schemas they specify, the query languages they support, and the terminologies they recognize. Heterogeneous database systems attempt to unify disparate databases by providing uniform conceptual schemas that resolve representational heterogeneities, and by providing querying capabilities that aggregate and integrate distributed data. Research in this area has applied a variety of database and knowledge-based techniques, including semantic data modeling, ontology definition, query translation, query optimization, and terminology mapping. Existing systems have addressed heterogeneous database integration in the realms of molecular biology, hospital information systems, and application portability.

Key Words: database; heterogeneous database; federated database; database integration; data warehouse.

Water, water, everywhere, Nor any drop to drink.
"The Rime of the Ancient Mariner," Samuel Taylor Coleridge

INTRODUCTION

The rapid expansion of biomedical knowledge, reduction in computing costs, and spread of internet access have created an ocean of electronic data. Today, databases around the world contain biomedical data ranging from the clinical findings for an individual patient to the genetic structure of our species. Many of these systems are connected or accessible via internet standards. In aggregate, these data encompass information and knowledge that can significantly improve patient care, public health, basic research, and administrative efficiency. However, the wonderful volume and availability of these data have grown through a largely decentralized process that has allowed organizations to meet specific or local data needs without requiring them to coordinate and standardize their database implementations. This process has resulted in a patchwork of diverse, or heterogeneous,

1532-0464/01 $35.00

database implementations, making access to and aggregation of data across implementations very difficult from a practical perspective. The practical problems of heterogeneous database integration create a large gap between the potential and the realized value of electronically stored data.

For example, many hospitals have information systems for their administrative, laboratory, pharmacy, ICU charting, and other functions, but very few have integrated physician workstations that allow clinicians to review data for a single patient across all of these functions. Also, numerous online resources now exist for molecular biologists that together characterize the genomic, proteinomic, and clinical features of many diseases. However, most biologists must learn several different user interfaces to access and cross-reference all these data, and many types of useful information requests are not possible, although the data to answer them are available.

The creation of software systems to overcome these problems may seem to be a simple matter of "programming." In actuality, very significant theoretical barriers impede the integration of heterogeneous data sources. The foremost of these barriers is the representational heterogeneity of the data themselves, that is, the differences in data models, schemas, naming conventions, and levels of abstraction used to represent data that are conceptually similar. Additional theoretical challenges include performance optimizations for translating queries and executing them across multiple databases, and methods to efficiently maintain mappings among databases that are autonomously managed and frequently changed.

In biomedicine and other domains, the problems of heterogeneous database integration are being addressed in research and application environments. This paper reviews the nature of heterogeneous database integration, the general methodologies that researchers have pursued to overcome the problem, and the specifics of several database-integration projects in biomedicine.

BACKGROUND

The process of heterogeneous database integration may be defined as "the creation of a single, uniform query interface to data that are collected and stored in multiple, heterogeneous databases." Several varieties of heterogeneous database integration are useful in biomedicine.

1. Vertical integration. The aggregation of semantically similar data from multiple heterogeneous sources. For example, a "virtual repository" that provides centralized access to mammography data that are collected and stored in databases across the United States [1].

2. Horizontal integration. The composition of semantically complementary data from multiple heterogeneous sources. For example, a system that supports complex queries across genomic, proteinomic, and clinical information sources for molecular biologists [2–4], or a physician workstation that provides a single interface to data stored in multiple ancillary systems [5–7].

3. Integration for application portability. The standardization of access to semantically similar information at disparate sources. For example, a universal database interface for decision-support applications that allows them to be shared across institutions with no modifications to their implementations [8].

Heterogeneous Database Systems

Heterogeneous database systems (HDBS) are computational models and software implementations that provide heterogeneous database integration [9]. For example, an HDBS might provide uniform access to electronic patient records in a hospital computing environment that uses a MUMPS hierarchical database for storing patient demographic data and a Sybase relational database for storing patient laboratory results. HDBSs are sometimes confused with distributed database systems (DDBSs) because both provide a unified view of and common interface to data that is physically stored in different locations. However, DDBSs are much more integrated and coordinated than are HDBSs. Typically, the constituent databases of a DDBS implement the same data model and query language and run the same distributed database management software. Also, the fragmentation of data in DDBSs is usually *induced* to reap the efficiency and autonomy advantages of distributed computing. The constituent databases in HDBSs, in contrast, existed prior to the establishment of the HDBS and are coordinated much more loosely. Specifically, HDBSs have the following characteristics [9]:

1. Representational heterogeneity. The constituent databases in an HDBS may use different data models, different query languages, different terminologies, and different schema structures to represent the same real-world semantics. In other words, although the data stored at multiple sites may have identical real-world semantics, the data representations and the data-access methods at the sites may differ.

2. Local autonomy. Each constituent database has the right to control access to its own data by the HDBS and

it has the ability to access and manipulate its own data independently of the HDBS. Most decisions regarding the representation and manipulation of data are made by local database administrators to accommodate local system needs (such as functionality, performance, and cost). The membership of the database in a heterogeneous database system is incidental to the primary purpose of the database.

3. Bottom-up integration. Whereas DDBSs induce the distribution of data that were previously integrated to achieve efficiency benefits, HDBSs integrate data that were previously distributed to achieve interoperability benefits. Bottom-up integration implies that the HDBS must provide an interface to diverse, preexisting information systems without requiring extensive modifications of preexisting software.

The goal of an HDBS is to provide database transparency to users and application programmers, that is, to provide a global and consistent database interface for applications, as if the data were not distributed and all of the database management systems were of the same type. HDBS research is predicated upon the belief that heterogeneity at the level of constituent database systems will (and should) persist, despite standardization efforts. This certainly is a realistic model for biomedical databases around the world in the foreseeable future.

Requirements for Heterogeneous Database Integration

It is useful to enumerate a set of requirements and assumptions to provide a context to the challenges of heterogeneous database integration. The following requirements are adapted from [10]:

1. Database heterogeneity is here to stay, at a variety of levels. Despite efforts and advancements in the area of standards, a single model for biomedical databases will not emerge.
2. Users and applications must be able to issue complex declarative multidatabase queries. Heterogeneous database systems must provide powerful and general query capabilities that retrieve all the information pertaining to a single object or all the objects that meet a set of search criteria. The capabilities should not be tied to any particular application or information need.
3. Users and applications should not be required to know the existence, physical location, access mechanism, or schema of the underlying local databases.
4. Write access to local databases is not required by most users and applications. The contents of the local databases may be maintained autonomously and locally.
5. The schemas of local database change quickly (on average of two or three times per year). The databases are designed and maintained to meet local needs, and changes are made independently of the integrated database structure.
6. Updates to local databases occur frequently, and value is placed on timely access to the newest data.

Query Models: A Framework for Considering Heterogeneous Database Integration

The core of the database-integration problem is that independently developed and maintained databases are heterogeneous with respect to their *query models*. Informally, a query model [8] is the model of data storage and information retrieval that must be known to a user or database programmer when she encodes the conceptual notion of an information request into the executable commands of a formal query language. Query models consist of the following four components:

1. The abstract *model* of data representation that applies to the database (for example, can the data be thought of as represented by unstructured text files, by relational tables, or by the tree structure of a hierarchical database?);
2. The *schema* of the specific data that are represented in the database (for example, are the names of patients and the names of the medications taken by those patients represented in a single file, which the query can directly access, or are they represented in different files, which the query must compare or join?);
3. The *language* for specifying queries that can be processed by the database, including syntax and semantics (for example, does the database require low-level commands that instruct it precisely where to find the names of all drugs associated with a patient name, or can it process high-level commands, such as SQL, that describe declaratively which patients and drugs to retrieve?);
4. The *format* of the data that are represented in the database (for example, are the names of drugs represented as the full trade names used by clinicians, as the abbreviated names used by the hospital formulary, or as the numerical codes used by a national drug-coding system?).

The task of heterogeneous database integration is to create a single “virtual” query model that encapsulates the query models of constituent databases and allows users and programs to access data from the constituent databases using this virtual model. Figure 1 illustrates the concept. All heterogeneous database systems, in biomedicine and other domains, provide this service, although the general strategies

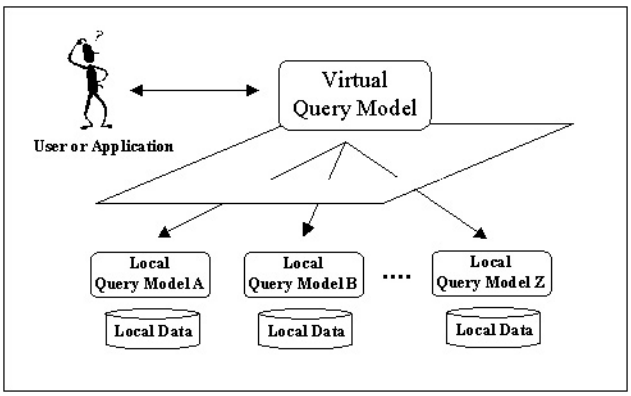


FIG. 1. The encapsulation of local query models provided by heterogeneous database systems.

and specific technologies that they employ differ significantly.

THE CRUX OF THE PROBLEM: REPRESENTATIONAL HETEROGENEITY

The largest barrier to heterogeneous database integration is the variety with which similar data are represented in different databases, i.e., representational heterogeneity. It is instructive to consider the several types of representational heterogeneity that schema integration techniques must resolve. The most general type of heterogeneity is that of the data models themselves. Aggregating data from relational, hierarchical, object-oriented, and flat file databases into a single representation is the first step in schema integration. However, even if all database systems were to use the relational model, significant representational heterogeneity would remain. Specifically, there exist structural differences, naming differences, semantic differences, and content differences. Examples of each of these differences are discussed below, with examples adapted from [11].

Structural Differences

Structural differences may consist of alternative table decompositions (horizontal and vertical), differences in data versus metadata representation, and differences in structured versus free-text encodings. Alternative horizontal table decompositions entail different degrees of normalization that result in the same information being distributed across a varying number of tables. For example, the relationship between a patient and a physician may be represented in at least two ways, as shown in Fig. 2.

Alternative vertical decompositions entail different distributions of rows among one or more tables. Rows may be partitioned in certain databases across multiple tables to improve retrieval performance when most (local) queries access only a subset of all the rows. For example, the representation of inpatient and outpatient records in the database of a large hospital may be represented in at least two ways, as shown in Fig. 3.

Because the relational model has no constructs for representing type hierarchies directly, such hierarchies may be encoded in a variety of ways in relational databases, and these encodings entail differences in the use of data and metadata. For example, the encoding of two types of serum electrolyte results, serum sodium and serum potassium, may be represented in at least three ways, involving the use of table names, field names, or field values, as shown in Fig. 4. Note that the alternative encodings represent exactly the same semantic information.

Structured versus free-text encodings are very common sources of heterogeneity and entail differences in the distribution of primitive data elements across multiple fields versus concatenated in one field. Common examples include the separation or concatenation of laboratory result values and units (i.e., ⟨“145 mg”⟩ versus ⟨“145,” “mg”⟩, and the separation or concatenation of address components (i.e., ⟨“125 Elm St., Denver, CO 80220”⟩ versus ⟨“125 Elm St.,” “Denver,” “CO,” “80220”⟩).

Naming Differences

Naming differences are characterized by distinct lexical terms denoting the same semantic objects across database schemas. Naming differences may be manifested as metadata differences or as data differences. Metadata differences are among the simplest forms of database heterogeneity and comprise variations in the names of tables and fields, such as “Doctor” versus “Physician” or “MRN” versus “Patient__ID.” The difficulty with metadata naming differences is discriminating differences that are solely syntactic from differences that represent variations in semantics. For example, “MRN” versus “Patient__ID” is a semantic rather than naming difference if “Patient__ID” denotes the social security number of the patient rather than the medical record number (semantic differences are addressed in the following section). Detecting these sometimes-subtle semantic distinctions is among the most time-consuming aspects of database schema integration.

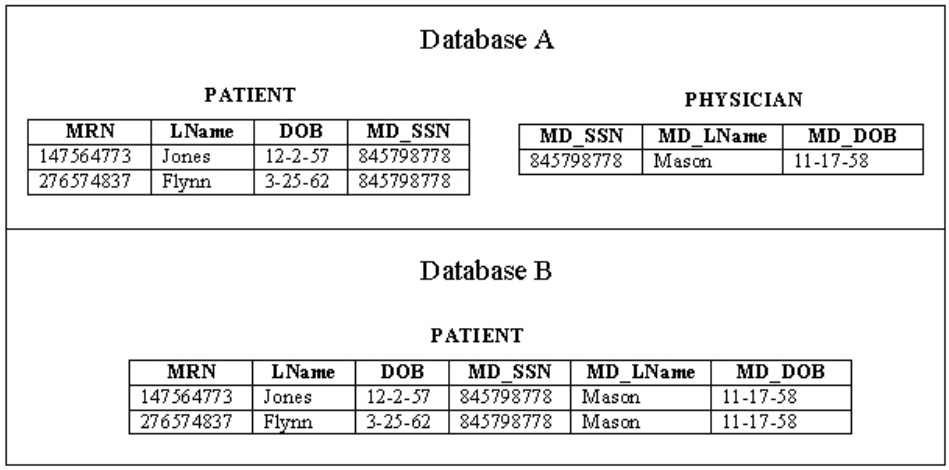


FIG. 2. Alternative horizontal table decompositions.

Similarly, data-naming differences are characterized by disparities among the symbols used to denote synonymous instances in heterogeneous databases. Examples include variations in the naming of diseases ("MI" versus "myocardial infarction") and test names ("Na" versus "Serum_Na" versus "Serum_sodium"). In the biomedical domain, where nomenclature is complex, sometimes ad hoc, and often overlapping, this "vocabulary problem" is a significant issue for any system that seeks to aggregate or compare data collected at distinct sites. A subfield of medical informatics is devoted to these terminology issues, and a large government-funded terminology resource now exists to assist in mapping synonymous terms to each other [12]. Many vocabulary problems in biomedicine, however, go beyond syntactic (naming) differences to the more difficult issue of semantic differences.

Semantic Differences

Semantic differences occur when the meanings of table names, field names, and data values across local databases are similar but not precisely equivalent. This problem is

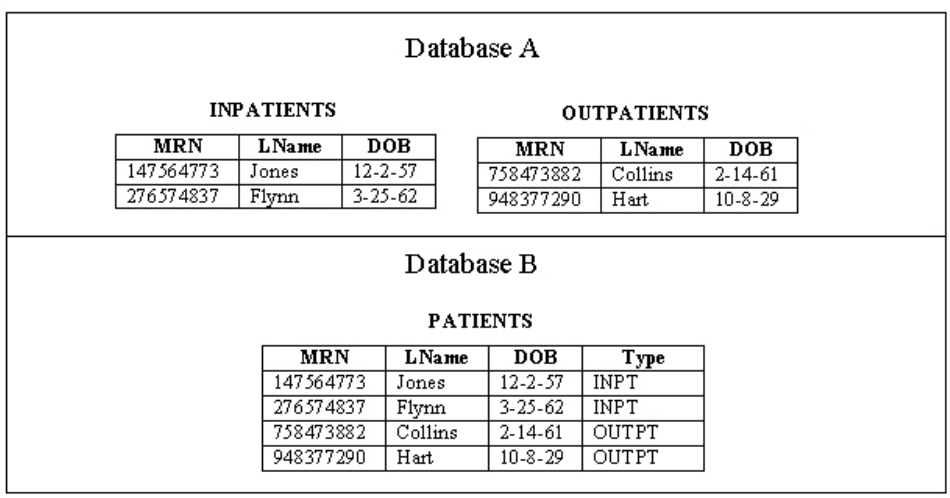
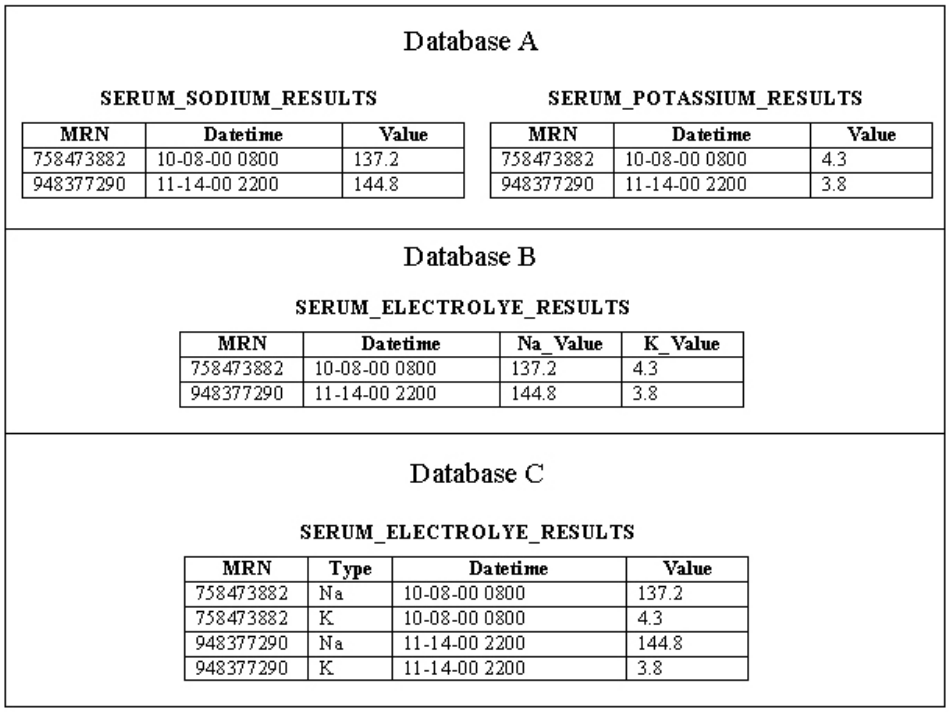


FIG. 3. Alternative vertical table decompositions.



Database A

SERUM_SODIUM_RESULTS

MRN	Datetime	Value
758473882	10-08-00 0800	137.2
948377290	11-14-00 2200	144.8

SERUM_POTASSIUM_RESULTS

MRN	Datetime	Value
758473882	10-08-00 0800	4.3
948377290	11-14-00 2200	3.8

Database B

SERUM_ELECTROLYE_RESULTS

MRN	Datetime	Na_Value	K_Value
758473882	10-08-00 0800	137.2	4.3
948377290	11-14-00 2200	144.8	3.8

Database C

SERUM_ELECTROLYE_RESULTS

MRN	Type	Datetime	Value
758473882	Na	10-08-00 0800	137.2
758473882	K	10-08-00 0800	4.3
948377290	Na	11-14-00 2200	144.8
948377290	K	11-14-00 2200	3.8

FIG. 4. Alternative encodings of a simple type hierarchy.

particularly insidious when the labels of tables, fields, and data values are identical across databases, but their meanings are, in fact, different [13]. To create a uniform query model that has well-defined semantics and produces accurate query results, these semantic differences must be recognized and resolved. Semantic differences may occur when there is no one-to-one correspondence among the *concepts* denoted by values within local databases.

For example, assume that two local databases each contains a field named BLOOD__CULTURE__GROWTH, which store values indicating the degree of growth observed for a culture specimen in the microbiology lab. Database A indicates the level of growth with the values "no__growth," "moderate growth," and "significant growth," whereas database B uses the values 0, 1+, 2+, 3+, and 4+. These values create a semantic mismatch in that there is a *one-to-many* correspondence between the value sets of database A and database B. A reasonable strategy to integrate these value sets would be to select the more general value set for the global schema ("no__growth," "moderate__growth," etc.), and to map the more specific values (0, 1+, 2+, etc.) to this set at the time data were exported or queries translated. This strategy would result in the following value mappings, which will support accurate query semantics at the global level:

Local values		Global values
0	→	no__growth
1+ or 2+	→	moderate__growth
3+ or 4+	→	significant__growth

Although this strategy of selecting the most general values is effective, it also results in loss of information as more databases are integrated. For example, if a third database were added that distinguished only between "growth" and "no growth," the granularity of data at the global level would be further reduced to just those two values, since they now represent the "least common denominator."

Additionally, cases arise in which semantic differences exist among values sets that cannot be resolved through mappings. Specifically, this occurs when there is an overlapping (many-to-many) correspondence between value sets. For example, assume the values in database A were

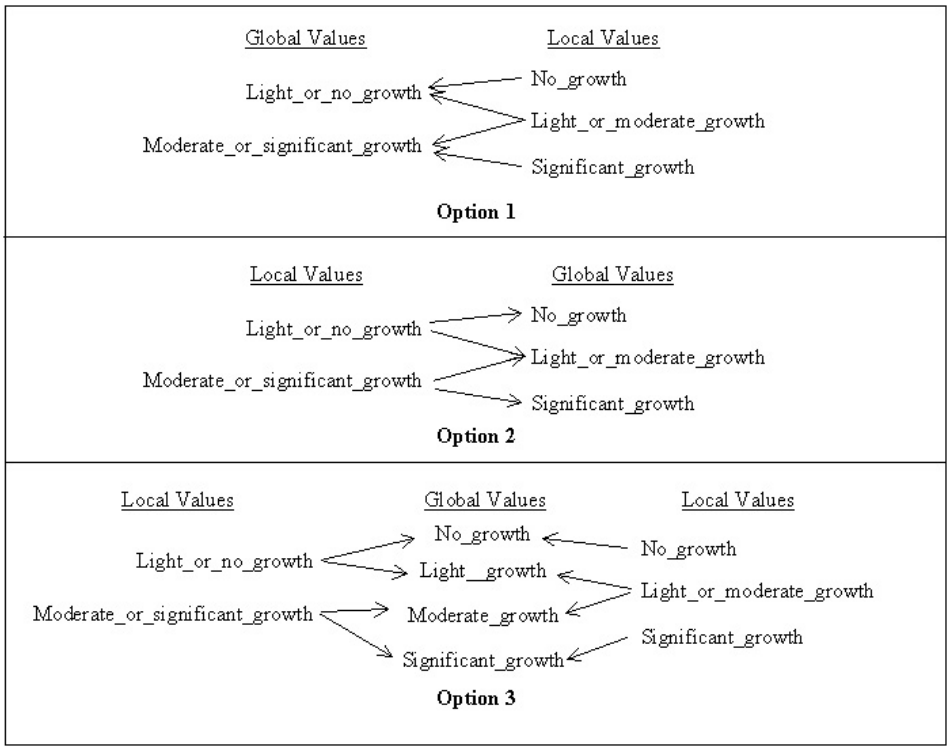


FIG. 5. Options for mapping test-result values, none of which resolve the semantic heterogeneity in this example.

"light__or__no__growth" and "moderate__or__significant__growth," and in database B "no__growth," "light__or__moderate__growth," and "significant__growth." In this case, no mapping between value sets exists that guarantees semantically accurate data transformations or query transformations. The reader may convince himself of this by examining the mapping options in Fig. 5. Note that none of the options result in a global value set that ensures that data arriving from all local databases correspond to one and only one of the global values. For example, in option 1, a query requesting only values indicating "light__or__no__growth" at the global level might, in fact, return an instance of moderate growth given the local values and the specified mappings. This example underscores the often-vexing nature of heterogeneous data integration, and the need for further research into resolving semantic data differences.

Content Differences

Content differences occur when data represented in one local database are not directly represented in another. The data may be implicit, derivable, or simply missing.

Implicit data are usually constant, and therefore assumed, within the environment of a local database, but cannot be assumed in the context of the global database. For example, an integrated database that provides a registry of all licensed physicians in the United States must explicitly represent the specialty and board certification of each physician represented. However, if the underlying local databases are the membership registries of specialty societies, the specialties and board certifications of stored individuals will not be represented because they are implicit. In these cases, the data-transformation or query-transformation processes will need to introduce values for the specialty and board certification depending on the source database.

A classic example of derivable data is the representation of zip code versus state or date-of-birth versus age. Clearly, each may be derived from the other (with some loss of information depending on the "direction" of the computation) [14]. These types of arbitrary transformations, however, illustrate the need for general-purpose functions within data-translation or query-translation software modules because declarative mappings are often not powerful enough to resolve such differences.

The problem of missing data obviously occurs when the global schema contains an information type that is simply not available in one or more local databases. For example, a specific clinical facility may choose to omit patients' HIV statuses from its electronic database for purposes of confidentiality, whereas an integrated database that tracks AIDS epidemiology may include "HIV status" as a field. Typically, a NULL value is denoted at the global level in these cases, but this convention is inadequate because the meaning of NULL values in databases is ambiguous [15]. In the case of HIV status, a NULL value may signify that the status is negative, the status is unknown, or the status is known but unavailable (the appropriate meaning in this case). Further research is required to accurately represent the semantics of missing information in the query models of heterogeneous databases.

COMPONENTS OF HETEROGENEOUS DATABASE SOLUTIONS

To provide uniform interfaces (query models) for heterogeneous databases, researchers in database and knowledge-based systems have pursued numerous strategies and focused upon several core issues.

Data Translation versus Query Translation

The most general distinction among heterogeneous database systems is whether they employ a data translation or query translation strategy.

Data translation involves the transformation of data from the various native formats in which they are collected and stored to a common shared format in which they can be uniformly accessed. The shared format directly implements all of the elements of a query model, enabling users and applications to ignore the specifications of the constituent query models. Data warehouses for diverse bioinformatics data [16] and clinical data repositories [5, 17] are examples of this integration strategy. Data translation may take place manually or automatically. Manual data translation, which is often used to aggregate clinical data into epidemiologic databases and disease registries, requires medical records personnel to manually abstract patient records into a format consistent with a standard clinical data set and with a standard coding scheme. Although this method certainly works, the costs and time requirements obviously inhibit the creation of many such integrated resources.

Automated data translation, which is more common, algorithmically translates data from the stored formats in which they are captured to the common format. Gateways may convert data directly to the shared format [5], or they may translate the data first to a common "interchange" format (such as HL7 [18]) that is standard across database implementations and that subsequently can be translated to the common format. Although automating data translation reduces the costs and delay of translating data, this strategy still suffers from two problems. First, data translation entails the duplicate storage of data in both the original and the shared format, which increases both the cost of operating an information system and the chances of compromising data integrity. Second, the correspondence between the stored format and the shared format is represented only in the encoded algorithms of the translating programs, and this "procedural" representation is difficult to inspect, validate, and maintain. A procedural representation of the correspondence between disparate data formats increases the chances of incorrectly translating complex medical data and increases the costs of maintaining the translation algorithms when the underlying databases change. Figure 6 illustrates the flow of information inherent in the data-translation strategy.

The alternative strategy for providing a virtual query model is to translate queries rather than data. In this strategy, the virtual query model is truly virtual, and data are stored only in the constituent heterogeneous databases. At the time a query is issued against the virtual model, a query-translation and query-execution engine decompose and translate the query into an equivalent set of local queries. The local

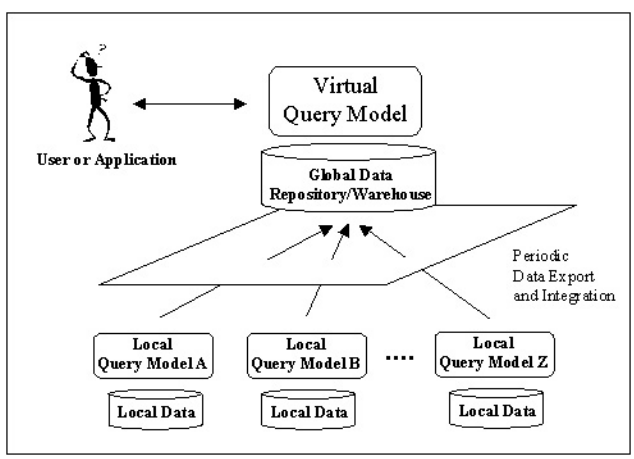


FIG. 6. The data-translation strategy for heterogeneous database integration.

queries are executed directly against the constituent databases, and the results are transmitted, transformed, and combined for presentation to the user or calling application. Figure 7 illustrates the flow of information inherent in the query-translation strategy.

The disadvantages of query-translation systems are that they entail additional performance overhead in the transformation and the remote execution of queries, that they are significantly more difficult to implement, and that they are not feasible when data sources lack ad hoc query interfaces. Nevertheless, given that many applications require timely access to real-time data and many heterogeneous databases cannot be compelled to periodically export data to a shared repository, the query-translation strategy is very valuable in many real-world settings.

Global Data Models and Query Languages

As shown by HDBS research, database interoperability requires the use of a common data model that is sufficiently simple and abstract to represent the contents of various heterogeneous data models and a corresponding query language that can be used to formulate queries at an equally abstract level. Semantic data models [19, 20] provide appropriately abstract conceptualizations of domain data that can serve as common modeling environments for disparately implemented databases. Although semantic data models (SDMs) have been traditionally used for database design because they provide more powerful abstractions for the specification of database schemas than are supported by the relational, hierarchical, and network models, the abstract and expressive modeling constructs of SDMs also are well suited for the specification of global conceptual schemas that model domain data in an implementation-independent way. Experimental HDBS systems typically have used simple semantic data models and set-oriented query languages, such as the functional model and the DAPLEX query language [21] and the entity–relationship model and the GORDAS query language [22]. Simple data models that represent "atomic" facts, such as entities and the relationships among entities, are easier to translate to the more complex data models that are implemented by constituent database systems. Set-oriented query languages, such as the relational algebra, provide greater potential for query decomposition and query optimization than record-at-a-time languages, such as CODASYL.

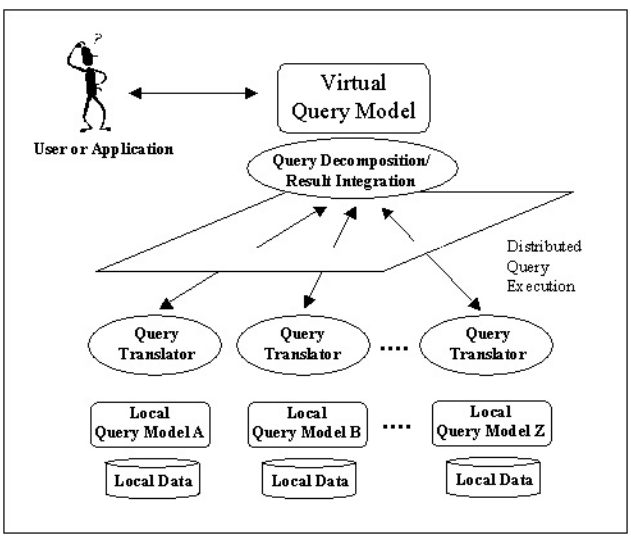


FIG. 7. The query-translation strategy for heterogeneous database integration.

The best-known examples of SDMs are the entity–relationship (ER) model [23], including numerous variations of it [24, 25], and the functional data model [26, 27]. Other SDMs include extended relational models [24, 28], hybrid ER and functional models [29], and data models based on semantic networks [30].

Although much of this research has occurred in the database community, knowledge-based researchers have also addressed the issue of global schemas for encapsulating heterogeneous system implementations. In the knowledge-based literature, these global conceptualizations are often called "ontologies" or "domain terminologies" rather than global schemas. Ontologies define the object classes, relationships, functions, and object constants for some domain of discourse [31]. An ontology is similar to a query model in that both include the following components:

1. A formal abstract model for representing the properties of objects in a domain;
2. A definition of the objects classes and of the relations and functions that may be defined over the members of those classes in a particular domain (the schema component of a query model);
3. A specification of the object constants that may be members of the defined object classes (the format component of a query model).

If an ontology also includes a query language, then it is indeed equivalent to a query model. Given the similarities, it is instructive to consider certain findings from ontology research in considering the design of global schemas for heterogeneous databases. Specifically, Gruber has specified a set of design criteria for ontologies that are intended to

support interoperability based on a shared conceptualization [32]:

1. Clarity. An ontology should effectively communicate the intended semantics to humans who formulate queries or design applications based on the ontology.
2. Coherence. An ontology should be internally consistent.
3. Extensibility. An ontology should support the addition of new concepts (to support the needs of a specific site or when general domain knowledge increases) without requiring revisions of the existing definitions (and, by implication, of applications that use those definitions).
4. Minimal encoding bias. An ontology should be specified at a sufficiently abstract level to enable it to encompass many different representation systems and styles.

Gruber's criteria provide a useful benchmark for the selection of data modeling formalisms and the design of global information schemas for heterogeneous databases. Indeed, a number of heterogeneous database projects have used ontology-specification languages borrowed from knowledge-based research to specifying global information schemas across data sources. The SIMS project [33], for example, uses the LOOM knowledge representation system to specify a global schema in the transportation domain, whereas the TAMBIS project [4] uses the GRAIL description logic to specify a global schema for bioinformatics databases.

Query Translation

HDBS researchers pursuing the query-translation strategy also have explored techniques to translate queries formulated in a global data-manipulation language to equivalent queries formulated in the specific data-manipulation languages of constituent databases. The software components that perform these functions have been termed "drivers" [3], "wrappers" [4], "mediators" [34], "site servers" [7], and "encapsulators." The techniques that these components use fall into two categories.

1. Query translation based on procedural mappings. In this method, the mappings between the conceptual database schema and various underlying database implementations are represented as procedural functions or programs that physically import objects stored in underlying databases into corresponding objects stored in the global environment, where they may be manipulated further as part of query processing. This approach characterizes the database-integration strategy of the Physician Workstation project at Hewlett–Packard Laboratories [6], and the Kleisli query system for integrating diverse bioinformatics resources [3]. This strategy is effectively a hybrid of the data-translation and the query-translation strategies: Procedural functions translate and import data from a stored format into the shared format, but the functions are only invoked when queries require data that the functions provide. Also, the mapping functions may specify that the underlying database not only retrieve requested objects, but also apply certain data operations included in conceptual-level queries (such as filtering results).

The advantage of query translation based on procedural mappings is that procedurally specified mappings can accommodate a wide range of legacy database implementations. The disadvantage of this approach is that the query engine cannot perform certain optimizations because the mappings between query models are specified as procedural functions that cannot be decomposed, reordered, and recombined to achieve maximum efficiency [6]. Also, procedural mappings are more difficult to maintain when the underlying data sources change because actual programs must be modified and retested.

2. Query translation based on declarative mappings. To maximize the potential for optimization and minimize the costs of maintenance, a declarative representation of query-model mappings is preferred. A declarative representation specifies the *correspondence* between objects and operations at the level of the global query model and objects and operations of the various constituent query models. The representation of the correspondence is (1) formally encoded such that a software process may inspect it, and (2) stored independently of the software code that actually performs query translation. Because a query optimizer can inspect and manipulate such declarative mappings based on an "understanding" of the mappings' semantics, the optimizer can apply information about the semantics of the data, the state of the local database, and the capabilities of the local DBMS to determine an optimal sequence of query operations. For example, an optimizer might discern that a request for two types of objects at the global level can be processed by a single query at the local level because both object types are stored in the same local table. Also, declarative mappings reduce the effort required to maintain the query-translation mechanism when changes to the schemas and other elements of the underlying query models occur. This is a significant concern in the practical operations of heterogeneous database systems because autonomous changes to the underlying databases are not infrequent. The disadvantage of declarative mappings is that they are less powerful in resolving query-model differences than procedural mappings, which can

bring the full power of Turing-complete programming languages to bear on the transformations required to resolve complex query-model differences. Indeed, most query-translation systems based on declarative mappings also provide the capability to introduce procedural functions to resolve the thorniest query-model mismatches.

Although query translation based on declarative mappings is less common than that based on procedural mappings, there are several systems that employ this method. The TransFER system [11] uses an extended version of the relational algebra to encode query-model mappings between a semantic data model and various relational database implementations. A translation engine applies the mappings (via an attribute grammar) to transform global queries to equivalent local SQL queries. A formal evaluation of this technique demonstrated that it provides significant power in mapping to a large variety of relational database implementations and allows extensive optimizations to improve the performance of the resulting local queries [35]. The Object Protocol Model multidatabase query system [2] also uses declarative mappings stored in a "metadata file" to translate queries specified against an object-oriented semantic data model to SQL queries against heterogeneous relational databases. Both of these systems translate queries by applying transformations to the syntax trees of the global queries to generate one or more equivalent local queries. The knowledge-base community has also implemented systems in which query-model mappings are specified declaratively using description logics. Query translation is performed in these systems via rule-based inference over the mappings [33]. The disadvantage of rule-based techniques relative to tree-transformation methods is that rule-based translation entails a heuristic search of the space of possible transformations, with possible backtracking. Tree transformations, conversely, support the deterministic modeling of transformation rules (in attribute grammars, for example), such that the rules are applied in a predetermined sequence, which avoids costly heuristic search and backtracking.

NOTABLE HETEROGENEOUS DATABASE SYSTEMS IN BIOMEDICINE

Several heterogeneous database integration challenges have commanded the attention of medical informatics researchers in recent years. Notable among these are (1) systems that provide molecular biologists seamless access to the growing number of bioinformatics databases, (2) clinical repositories that aggregate information from ancillary systems in hospital settings, and (3) database abstractions that insulate clinical decision-support applications from heterogeneous database implementations at various healthcare institutions.

Molecular Biology

The vast and complex compendium of molecular biology knowledge is available today in electronic databases, often accessible via the internet. These databases store DNA sequences (GenBank, GDB), protein sequences (Swiss-Prot), protein 3D structures (PDB), gene mutations (OMIM), enzyme activity (ENZYME), and many other types of information [36]. The ready availability of these data has undoubtedly accelerated the pursuit of basic research, the study of diseases, and the development of new medications. However, because these databases were developed independently and are managed autonomously, they are highly heterogeneous, difficult to cross-reference, and ill-suited to processing open-ended queries [37]. This heterogeneity limits the ability of molecular biologists to answer ad hoc queries involving multiple databases, such as "return all mammalian gene sequences for proteins identified as being involved in intracellular signal transduction."

Researchers are pursuing several avenues to overcome these limitations. Some projects have implemented data warehouses that physically aggregate and integrate heterogeneous data sources within a single database management system [14]. This data-translation approach provides excellent query response time, but may encounter limitations as the number and size of molecular biology databases grow and the maintenance challenges of uploading local updates increase in complexity [38].

Other researchers are pursuing the query-translation approach and providing uniform query models to physically distributed data sources. The Bio-Kleisli project [3] uses a powerful functional language to specify procedural mappings among query models and to integrate heterogeneous query results. This project is notable in that it has successfully implemented several "impossible" queries, as posed in an informatics summit of the human genome project in 1993 [39]. However, Bio-Kleisli does not provide a global schema of molecular biology data and, therefore, still requires users to know a great deal about the contents and structure of underlying data sources. The Object Protocol Model (OPM) system [2] defines an object-oriented semantic data model to encapsulate multiple data sources and applies declarative mappings and formal query-translation techniques to process

multi-database queries. The formalisms of the OPM provide more opportunities for query optimization, but are practically limited in that they require the specification of Object Protocol subschemas for each participating information source, which may be prohibitive.

Whereas these systems primarily apply techniques from the field of databases and programming languages, the TAMBIS project supports query formulation over diverse information sources by applying knowledge-based techniques [4]. These techniques include the specification of a global ontology for molecular biology using description logic [40], as well as the use of logical concept definition for specifying queries. These techniques are interesting in that they support an intuitive user interface for navigating the global schema (ontology) and for graphically formulating queries.

Hospital Information Systems

Hospitals are notorious for containing "islands" of information across various departments, which are difficult to access separately or to integrate reliably. Clinical practice in hospitals could benefit greatly from the integration of these information islands, but the heterogeneity of the departmental information sources often impedes this. Many efforts have been made to overcome this difficulty.

Most hospital data-integration efforts entail the development of a physically separate database that aggregates data from the various departmental systems and makes them available in a "clinical data repository" for online access, decision support, and reporting. This data-translation strategy has historically required hospitals to laboriously construct and painstakingly maintain custom interfaces for each departmental system [5]. Increasingly, the HL7 messaging standard is used to simplify the creation and maintenance of such interfaces [41]. However, either data-translation approach suffers from the limitation that data are not available in the global repository until they are physically updated from the departmental systems, a significant limitation for certain clinical applications.

A few integration efforts in hospital environments have pursued the query-translation approach to provide real-time access to data in ancillary systems. The Physician Workstation project [6] defines an object-oriented reference schema and translates high-level queries specified against this schema to operations against the underlying databases. The procedural mappings used in this translation process, however, limit the opportunity for query optimization, a disadvantage in real-time clinical environments (although this query-translation approach still provides more timely access than the data-translation alternative). W3-EMRS [7] is a clinical-data integration project that dynamically integrates clinical information from departmental systems and displays them in a web interface. The system defines a global schema called the "Common Medical Record" and implements a software module (the "agglutinator") that broadcasts queries against this schema to all participating information systems. A "site server" at each of these systems translates the queries to the local data-manipulation language, executes the queries, and transmits the results back to the agglutinator. W3-EMRS provided an effective web-based interface to heterogeneous data for clinicians, but is limited in that the site servers only perform query translation for data objects that are predefined, limiting the ad hoc querying capabilities of the system.

Application Portability

Decision-support applications in medicine require patient-specific data to provide advice in the context of clinical cases. Examples include diagnostic systems, such as Mycin [42], and clinical event monitors, such as Medical Logic Modules [43]. The emergence and expansion of clinical databases provide the potential for decision-support applications to access patient-specific data automatically. Because the means by which data are represented and retrieved vary widely among clinical databases, however, decision-support applications that automatically access patient data currently cannot be shared easily among healthcare institutions, which limits their widespread adoption. Sujansky developed a system called TransFER [11] that provides decision-support applications with a uniform interface to clinical data stored in heterogeneous relational databases and facilitates the sharing of such applications across institutions.

The TransFER methodology supports the definition of a global reference schema of clinical information against which applications may formulate requests for clinical data. The reference schema is specified using a semantic data model ("FER") that is a hybrid of the functional data model and the entity–relationship data model. The data model includes a declarative query language ("Refer") and a formal mapping language, based on the relational algebra, which allows database administrators to specify the correspondence between the global reference schema and the local relational database at their site. A translating compiler at each site uses the encoded mappings to translate automatically queries that are specified against the global schema to semantically equivalent queries that can be executed by the local relational database.

Sujansky evaluated a prototype implementation of the TransFER methodology, which entailed mapping a reference schema of patient data to three distinct clinical databases [35]. The evaluation demonstrated that (1) the FER data model and Refer query language are well-suited for representing clinical database queries, (2) the mapping language is effective at mapping FER schemas to heterogeneous relational database schemas, and (3) the translating compiler is capable of correctly translating Refer queries to equivalent SQL queries. The primary shortcoming of the TransFER approach is that automatically generated query translations result in less efficient queries than manually generated translations, but query optimization enhancements based on the formal properties of the mapping language hold promise for eliminating the difference.

SUMMARY

Heterogeneous database integration is a difficult but important problem in biomedicine. The decentralized nature of our scientific communities and healthcare systems has created a sea of valuable but incompatible electronic databases. These databases are highly heterogeneous with respect to their query models, i.e., the data models they employ, the data schemas they specify, the query languages they support, and the terminologies they recognize. Specifically, the various forms of representational heterogeneity across these databases frustrate efforts to integrate them. Heterogeneous database systems attempt to unify these disparate databases by providing uniform conceptual schemas that resolve representational heterogeneities, and by providing querying capabilities that aggregate distributed data (through data-translation or query-translation techniques). Although most heterogeneous database systems in biomedicine are in the research stages, these technologies hold promise for bringing the voluminous, widespread, and varied data that now exist together to satisfy our thirst for information.

REFERENCES

1. Ohno-Machado L, Boxwala AA, Ehresman J, Smith DN, Greenes RA. A virtual repository approach to clinical and utilization studies: application in mammography as alternative to a national database. In: Proceedings American Medical Informatics Association Fall Meeting: Hanley & Belfus, 1997; 369–73.
2. Chen IA, Kosky A, Markowitz VM, Szeto E. OPM*QS: the object-protocol model multidatabase query system. Technical Report LBNL-38181, Lawrence Berkeley National Laboratory, 1995.
3. Wong L. Kleisli, a functional query system. J Funct Program 1998; 1(1):1–38.
4. Stevens R, Goble CA, Paton NW, Bechhofer S, Ng G, Baker P, Brass A. Complex query formulation over diverse information sources using an ontology. In: Bornberg-Bauer E, De Beuckelaer A, Kummer U, Rost U, editors. Workshop on Computation of Biochemical Pathways and Genetic Networks, European Media Lab (EML), August 1999; 83–8.
5. Marrs KA, Steib SA, Abrams CA, Kahn MG. Unifying Heterogeneous Distributed Clinical Data in a Relational Database. In: Safran C, editor. Proceedings of the Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1994; 648–55.
6. Annevelink J, Young CY, T. P.C. Heterogeneous database integration in a physician workstation. In: Clayton PD, editor. Proceedings of the Fifteenth Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1992; 368–72.
7. van Wingerde FJ, *et al.* Using HL7 and the World Wide Web for unifying patient data from remote databases. Proceedings American Medical Informatics Association Fall Meeting: Hanley & Belfus. 1996; 643–7.
8. Sujansky W, Altman R. Toward a standard query model for sharing decision-support applications. Proceedings American Medical Informatics Association Fall Meeting 1994; 325–31.
9. Sheth AP, Larson JA. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput Surv 1990; 22(3):183–236.
10. Karp PD. A strategy for database interoperation. J Comput Biol 2(4):573–86.
11. Sujansky W. A formal model for bridging heterogenous databases in clinical medicine. Ph.D. thesis. Stanford University, 1996.
12. http://www.nlm.nih.gov/research/umls/umlsmain.html.
13. Kohane IS, Wingerde FJ, *et al.* Sharing electronic medical records across multiple heterogeneous and competing institutions. Proceedings American Medical Informatics Association Fall Symposium; Hanley & Belfus, 1996; 608–12.
14. Sweeny L. Guaranteeing anonymity when sharing medical data, the Datafly system. Proceedings American Medical Informatics Association, Fall Symposium: Hanley & Belfus, 1997.
15. Date CJ. Three-valued logic and the real world. InfoDB, Winter, 1989.
16. Ritter O, *et al.* Prototype implementation of the integrated genomic database. Comput Biomed Res 1994; 27:97–115.
17. Johnson SB. Generic data modeling for clinical repositories. J Am Med Inform Assoc 1996; 3(5):328–39.
18. Rishel W. Pragmatic considerations in the design of the HL7 protocol. In: Kingsland LC, editor. Proceedings of the Thirteenth Symposium on Computer Applications in Medical Care; IEEE Computer Society Press, 1989: 687–90.
19. Hull R, King R. Semantic database modeling: survey, applications, and research issues. ACM Comput Surv 1987; 19(3).
20. Peckham J, Maryanski F. Semantic data models. ACM Comput Surv 1988; 20(3):153–89.

21. Smith JM, Bernstein PA, Dayal NG, Landers T, Lin KWT Wong E. Multibase—integrating heterogeneous distributed database systems. In: Gupta A, editor. Integration of information systems: bridging heterogeneous databases. New York: IEEE Press, 1986; 163–75.
22. Ceri S, Pelagatti G. Heterogeneous distributed database systems. In: Ceri S, Pelagatti, G, editors. Distributed databases, principles and systems. New York: McGraw-Hill, 1984; Chap 15.
23. Chen P P-S. The entity–relationship model—toward a unified view of data. ACM Trans Database Syst 1976; 1(1):9–36.
24. Elmasri R, Wiederhold G. Data model integration using the structural model. In: Proceedings of ACM-SIGMOD International Conference on Management of Data. 1979.
25. Engels G, Gogolla M, Hohenstein U, Hyulsmann K, Lohr-Richter P, Saake G, and Ehrich H-D. Conceptual modelling of database applications using an extended ER model. In: Proceedings of the Ninth International Conference on Data and Knowledge Engineering. Amsterdam: Elsevier Science, 1992; 157–204.
26. Shipman D. The functional data model and the data language DAPLEX. ACM Trans Database Syst 1981; 6(1):140–73.
27. Wilkinson K, Lyngbaek P, Hasan W. The Iris architecture and implementation. IEEE Trans Knowledge Data Eng December 1989.
28. Codd EF. Extending the database relational model to capture more meaning. ACM Trans Database Syst 1979; 4(4):393–434.
29. Mylopoulos J, Bernstein PA, Wong HKT. A language facility for designing database-intensive applications. ACM Trans Database Syst 1980; 5(2):185–207.
30. Sowa J. Conceptual structures: information processing in mind and machine. Reading, MA: Addison-Wesley, 1984.
31. Fikes R, Cutkosky M, Gruber T, van Baalen J. Knowledge sharing technology project overview. Technical Report KSL 91-71, Knowledge Systems Laboratory, Stanford University, 1991.
32. Gruber TR. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-4, Knowledge Systems Laboratory, Stanford University, 1993.
33. Arens Y, Chee CY, Hsu C, Knoblock CA. Retrieving and integrating data from multiple information sources. Int J Intell Coop Inform Syst 1993; 2(2):127–58.
34. Aymard S, *et al.* Toward interoperability of information sources within a hospital intranet. Proceedings American Medical Informatics Association Fall Meeting; Hanley & Belfus, 1998; 638–42.
35. Sujansky W, Altman R. An evaluation of the TransFER model for sharing clinical decision-support applications. Proceedings American Medical Informatics Association Fall Symposium: Hanley & Belfus, 1996; 468–72.
36. http://molbio.info.nih.gov/molbio/db.html.
37. Markowitz VM, Ritter O. Characterizing heterogeneous molecular biology database systems. J Comput Biol 1995; 2(4).
38. Davidson SB, Overton C, Buneman P. Challenges in integrating biological data sources. J Comput Biol Winter 1995; 2(4):557–72.
39. http://www.bis.med.jhmi.edu/Dan/DOE/whitepaper/contents.html.
40. Schulze-Kremer S. Ontologies for molecular biology. In: Proceedings of the Third Pacific Symposium on Biocomputing: AAAI Press, 1998; 693–704.
41. www.hl7.org.
42. Shortliffe E. Mycin: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Ph.D. thesis, Stanford University, 1974.
43. Pryor TA, Hripcsak G. Sharing MLMs: an experiment between Columbia–Presbyterian and LDS Hospital. In: Safran C, editor. Proceedings of the Seventeenth Symposium on Computer Applications in Medical Care; New York: McGraw–Hill, 1994; 399–403.