Managing Heterogeneous Information Systems through Discovery and Retrieval of Generic Concepts

Uma Srinivasan*
CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde NSW 1670, Australia, Building E6B, Macqaurie University Campus, North Ryde NSW, Australia. E-mail: Uma.Srinivasan@cmis.csiro.au

Anne H.H. Ngu
School of Computer Science and Engineering, University of New South Wales 2052, Sydney, NSW, Australia. E-mail: anne@cse.unsw.edu.au

Tom Gedeon
School of Information Technology, Murdoch University, South Street Murdoch, 6150, WA. Australia. E-mail: tgedeon@murdoch.edu.au

Autonomy of operations combined with decentralized management of data gives rise to a number of heterogeneous databases or information systems within an enterprise. These systems are often incompatible in structure as well as content and, hence, difficult to integrate. Depsite heterogeneity, the unity of overall purpose within a common application domain, nevertheless, provides a degree of semantic similarity that manifests itself in the form of similar data structures and common usage patterns of existing information systems. This article introduces a conceptual integration approach that exploits the similarity in metalevel information in existing systems and performs *metadata mining* on database objects to discover a set of concepts that serve as a domain abstraction and provide a conceptual layer above existing legacy systems. This conceptual layer is further utilized by an information reengineering framework that customizes and packages information to reflect the unique needs of different user groups within the application domain. The architecture of the information reengineering framework is based on an object-oriented model that represents the discovered concepts as customized application objects for each distinct user group.

Introduction

Around the world, organizations have developed many database management systems or information systems to collect, store, and manage vast amounts of data. With the increase in demand for obtaining meaningful information from multiple information systems, it is essential to support some form of integration without disturbing the local autonomy of individual databases. The growth in the number and size of decentralized information systems within an enterprise has made the problem of integration more complex and challenging.

Initial research in heterogeneous databases was largely directed towards the issues of resolving schema and data heterogeneity conflicts across multiple autonomous databases (Batini, Lenzerini, & Navathe, 1986; Kim & Seo, 1991) and of developing a global schema to provide integration (Ahmed et al., 1991; Collet, Huhns, & Shen, 1991). Such an approach to schema integration has proved to be difficult, however, because these methods do not adequately capture the semantics represented by the schemata of the component databases. The difficulties encountered in achieving schema integration have highlighted the importance of capturing the semantics embedded in the underlying schemata. Volume 20 of *SIGMOD Record* (Segev & Sheth, 1991) is devoted entirely to the semantic issues of integration. A more recent volume of *SIGMOD Record* (Ouksel & Sheth, 1999) is also devoted to this issue in the context of data integration and retrieval in the World Wide Web. As we move away from well-understood structural aspects to the less well-understood semantics of databases, we need to address issues such as the meaning and use of data, where the different applications, database administrators, and end users have different contextual interpretations of the data. These issues have led to a series of international workshops on semantics of interoperable systems (Hsiao, Neuhold, & Sacks-Davis, 1992; Meersman, 1997). There is

*To whom all correspondence should be addressed.

a general agreement that integration can be achieved only with a good understanding of the embedded semantics of the component databases. Understanding semantics that are not explicitly represented requires *intelligence* in the form of domain knowledge or background knowledge of the application (Wiederhold, 1996). This knowledge is often not explicitly represented in database systems. It is embedded at various levels, for example, in the database model, data structures, constraints on the data, and in data domains not explicitly stated but understood by the user community. The major problem that needs to be addressed, therefore, is how to discover this knowledge from existing legacy systems to achieve database integration.

Knowledge discovery from databases is a recent research area that uses techniques from the field of artificial intelligence and machine learning. To be useful in the heterogeneous database arena, knowledge discovered from heterogeneous databases must be used to achieve some form of *intelligent integration* (ISX Corporation, 1994) or some form of intelligent cooperation of heterogeneous information systems (Brodie, 1992; Papazoglou, Laufmann, & Sellis, 1992). Recent knowledge of discovery methods use the data mining approach to discover knowledge, or concepts from databases, by looking for meaningful patterns in data (Piatetsky-Shapiro & Frawley, 1991). As pointed out by Wiederhold (1996), however, the major barrier for obtaining high-quality knowledge from data is due to the limitations of the data itself. Data is rarely collected for mining, and is often incomplete, and may have limited breadth or coverage.

Keeping in mind the limitations of the data in the data-mining approach, this article proposes a discovery method that uses existing database structures in the context of their use, or in other words, *metadata*, to automatically discover a set of generic concepts common to several databases or information systems in a given application domain.

In addition to the problems with embedded semantics discussed above, we need to address the difficulty that users have in retrieving information from multiple heterogeneous information systems/databases. Standard multidatabase query languages (Litwin, 1989) require precise specification of the data to be accessed, and methods to manipulate that data, and are therefore, not easily adaptable to a large, dynamic, autonomous environment. Other multidatabase query languages such as MDSL (Litwin & Abdellatif, 1986) and IDL (Krishnamurthy, Litwin, & Kent, 1991) extend the relational query model with new language features to specify semantically equivalent data elements, with an intention to resolve schematic differences across databases. These languages expect the users to have a good understanding of the data, and do not address the problem that users have in identifying semantically equivalent data. Most importantly, query languages package information in a similar fashion for all users regardless of their domain and contextual knowledge. This article suggests a solution to this problem in heterogeneous databases through the idea of *information reengineering*, where information is customized to reflect the domain knowledge of distinct user groups.

In summary, two main contributions are presented in this article.

(1) A concept discovery approach called ConceptDISH (Concept Discovery from Heterogeneous databases), to automatically discover generic concepts common to several existing databases/information systems within the same application domain. ConceptDISH exploits similarities in database structure and usage patterns to automatically discover a set of well-understood domain-dependent generic concepts that coincide with the users' vocabulary of the application domain. Once discovered, these generic concepts form part of the ontology of the application domain and serve as the basis for accessing information from heterogeneous databases.
(2) An information reengineering framework called ConceptVIEW to provide an information retrieval facility that packages or reengineers information from heterogeneous systems to provide a customized presentation layer that uniquely reflects the contextual need of distinct user groups. ConceptVIEW models the concepts discovered by ConceptDISH as application objects that reflect the representational vocabulary of different user groups within the application domain.

The conceptual integration approach proposed here focuses on developing a domain abstraction that serves as the basis for retrieving information from heterogeneous information systems within the same application domain. Complex object types and concept hierarchy are not considered in this approach to conceptual integration, as we are not attempting a translation or total transformation of the entire application domain. The next section describes the conceptual integration framework. Section 3 describes ConceptDISH—the concept discovery approach proposed to discover generic concepts common to several information systems within the same application domain. This section includes an empirical study that provided some valuable guidelines for the discovery process. Section 4 describes ConceptVIEW—the information reengineering framework that uses the discovered concepts to customize or package information such that users have the facility to access and view information using familiar concepts. Section 5 presents the related works, and finally in Section 6, we present the conclusions.

Conceptual Integration Framework

In the heterogeneous database context, the search for common concepts poses two substantive problems: first, the usual problems of data semantics and domain mismatch (Siegel & Madnick, 1991; Saltor, Castellanos, & Garcia-Salace, 1992); and second, the confidentiality of data (in this case patient clinical data) pose difficult ethical and social issues. The approach proposed in this article solves the data confidentiality problem by using metadata rather than actual data to achieve conceptual integration.

In the domain of clinical information systems—the application domain used throughout this article to demonstrate the techniques/approaches developed—there is sufficient empirical evidence to believe that users with a common objective store and manage similar types of data, even though the databases used may be heterogeneous. This happens because the basic specification for each individual database is constructed by users (e.g., doctors) who are equipped with similar domain knowledge. By the same token, this domain knowledge also plays a part while retrieving information from multiple databases. This leads one to believe that: (a) similar domain knowledge combined with common objectives leads to the design of similar database structures in heterogeneous databases, and (b) the usage pattern of legacy systems carries a cognitive load that indicates the background knowledge of users.

The conceptual integration approach proposed here exploits the background knowledge available in database structures and usage patterns and provides a "wrapping service" that generates a conceptual layer above existing legacy systems. This conceptual layer is further utilized by an information reengineering framework that packages information to reflect the unique needs of distinct user groups. While most "wrappers" transform data or translate metadata from legacy systems, this article proposes a domain abstraction approach to provide this conceptual middle layer. This middle layer is made up of common concepts automatically discovered from similar patterns in structure and usage of existing legacy systems.

Concept Discovery

A typical knowledge discovery system takes inputs in the form of raw data and produces outputs in the form of interesting patterns that can be classified in a concise manner. Once discovered, these pattern classes must be described such that they represent meaningful concepts. Discovering concepts requires application domain knowledge. When dealing with legacy databases, knowledge regarding the form and content of data, constraints on the data, and the application domain(s) described by the database represent the domain knowledge or background knowledge of the database. In database management systems, the data dictionary associated with the database stores information about the field names, data types allowed for the fields, and various other constraints associated with the data fields. The data dictionary defines the syntax of the database, or in other words stores metadata about the data that is managed in the database. Hence, the discovery approach presented here makes use of the data dictionary as a main source of application domain knowledge.

ConceptDISH uses a discovery algorithm that can be classified as learning from observation, where database objects of legacy systems are classified into groups or clusters that can be described by a concept from a predefined concept class that is well understood within the application domain. To establish a cluster of objects, each object (which is nothing but an entity as defined in the legacy system) is characterized by a set of variables. The variables represent the metadata of the object, and are good indicators of both structure and usage of the database object. Commonality in structure and usage pattern serves to establish similarity measures among objects that are clustered. The ConceptDISH algorithm partitions these objects into clusters using the values of the variables. The clusters thus formed are meaningful such that each cluster actually represents a concept common to different legacy systems in the application domain. The set of application specific generic concepts discovered in this way provides a domain abstraction and constitutes the reconstructed conceptual schema that serves as common ontology model to facilitate information retrieval from heterogeneous databases.

Information Retrieval

Research on information retrieval from heterogeneous systems has focussed on multidatabase query languages (Krishnamurthy, Litwin, & Kent, 1991; Litwin, 1989). Although all these approaches support interoperability, and ensure representational compatibility, research leading to this article has focussed on the issue of contextual differences among different user types while interpreting data from heterogeneous databases. The inherent differences among different user types often result in different contextual interpretation of the same piece of information. These differences can broadly be classified into three types:

(1) Differences in semantics: this occurs when the same term is interpreted differently by different users. For example, the *Nurse* user type might interpret the term (or symbol) *Diet* as a patient's diet requirement, while the *Dietitian* user type may regard the same term as the patient's *Treatment*. The difference here is not merely symbolic; it also represents a basic difference in semantic interpretation by the two user types.
(2) Differences in abstraction level: this occurs due to the inherent differences in information content required by different user types. For example, to treat a patient, the user type *Nurse* may need only the name and potency of the drug and the dosage frequency, while the *Oncologist* user type would possibly need to know the full details of the patient's treatment program. On the other hand, the *Pharmacist* user type may just need information about the number of chemotherapy patients being treated with a particular drug, as a guideline for reordering the drug. This indicates that although the source of information may be the same, the level of detail required may be different for different user types.
(3) Differences in association: this is governed by a user's perception of information content in a particular context. For example, the term *Diagnosis* could signify information like problem location, CAT scan reports, and tumor size to a *Surgeon* in the Oncology department. However, an *Oncologist* may associate the same

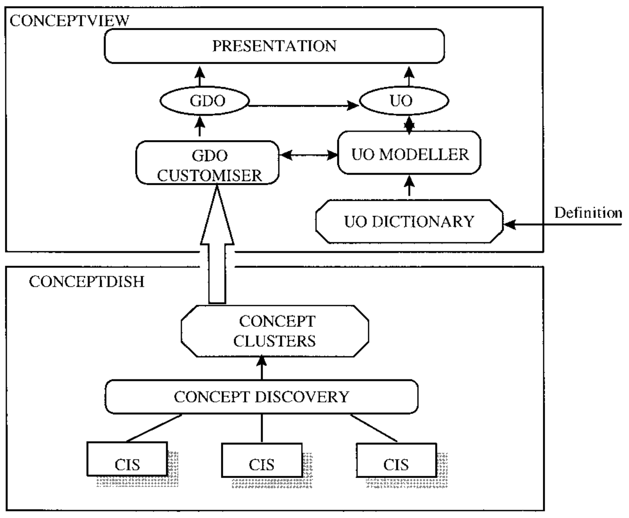

FIG. 1. System architecture.

term *Diagnosis* with information associated with details of oncological treatment, drug dosage monitoring, and so on. That is, the nature of a user's specialization (or background knowledge) has a direct impact on the path that is traversed to retrieve the required information. This is represented by the association links between the data and the symbols that represent them. (This is different from the semantic associations mentioned in the first point.)

The above differences suggest that it is impractical to force these disparate user types to use identical sets of objects for information retrieval. It is necessary to add some intelligence to the external view such that the information presented is meaningful in the context of the search, and adheres closely to concepts with which each user type is familiar.

The ConceptVIEW framework proposed for information retrieval models the concepts discovered by ConceptDISH as application object classes. ConceptVIEW acts as a "facilitator" in accessing, abstracting, routing, and transforming data as proposed in the reference architecture for intelligent integration of information [ISX Corporation, 1994, 1995). ConceptVIEW provides an external presentation layer that is unique to each user group to reflect the preferred terminology of that particular user group.

System Architecture

An overview of the system architecture that links ConceptDISH and ConceptVIEW is shown in Figure 1. The rectangular boxes show the main functions. The octagonal boxes show the main resources used for information packaging, and the oval shapes show the objects that help in implementing the architecture.

The ConceptDISH approach uses a concept discovery algorithm that classifies objects from existing clinical information systems (CISs) into generic concept clusters (shown by the octagonal box). Each concept cluster has a set of associated entities belonging to the CISs that participate in the clustering process.

The ConceptVIEW framework uses the concept clusters generated from ConceptDISH, and customizes each concept for each user type. The GDO (Group Data Object) customizer and User object (UO) modeler help to create a generic concept as application specific objects that reflect the domain knowledge of distinct groups of users. The UO modeler is supported by a user-object dictionary, which is initially created by allowing different user types to define their preferred terminology in the domain application. In conjunction with the UO modeler, the GDO customizer generates a set of group data objects (GDOs) for each user type represented by a User Object (UO). The presentation layer displays appropriate external labels for GDOs that are unique to each user type.

The ConceptDISH Approach

The concept discovery approach presented in this section uses the current structure of a part of a CIS together with usage pattern data to "discover" a set of concepts common to several CISs, and which are well known to the users of that application domain. ConceptDISH uses the similarity in static and dynamic properties of databases to discover concepts common to several cooperating databases/information systems in a given application domain. Static properties presented by schema descriptions and dynamic properties presented by usage patterns of different user groups form the kernel from which inductions are made about concepts common to several cooperating databases/information systems. An outline of this approach is presented in (Srinivasan, Ngu, & Gedeon, 1997). This concept discovery approach can be classified as learning from observation, where objects are classified into groups or clusters that can be described by a concept from a predefined concept set.

Objects and Variables

To establish a cluster of objects, each object (which is nothing but an "entity" as defined in a database) is characterized by a set of variables. The variables are chosen such that they are good indicators of both structure and usage of the database objects, and also serve to establish similarity measures among the objects that are clustered. A clustering algorithm partitions these objects into distinct clusters determined by the values of the associated variables.

The following variables are measured for each object to be clustered:

(1) *number of tuples* associated with the entity at a given point of time;
(2) *number of attributes*;
(3) *number of keys/links* to other entities;
(4) *usage frequency* at the time of recording the number of tuples;

(5) *number of users*;
(6) *number of user types*.

Although it may be possible to identify more variables for concept classification, this approach uses only those that are measurable in legacy systems.

Motivation for Choice of variables

While designing an information system, schema design (which is a static property of a database) is influenced by both the user and the designer. In designs driven predominantly by the designer, structure and design gain precedence, and the entities are described such that the overall schema is in 3NF. On the other hand, in designs where entity descriptions are completely specified by users, the entity descriptions are often complete in themselves, but the overall schema often has redundancies and is not always in 3NF. With increased usage, both types of schema definitions described above undergo some changes to accommodate users' needs and to improve database performance. In a normalized schema, if frequent queries require too many join operations, the schema is often denormalized to improve system performance. In such cases, the newly formed entity is often described with the combined attributes of the original entities, thereby reflecting a more complete, commonly used concept. As a result, in most systems over a period of time, the underlying schemata evolve with users' varying information needs and perception of familiar concepts. The schema evolution, in turn, leads to the evolution of similar concepts as well as similar description of concepts across heterogeneous systems in a given application domain. The similarity is reflected through the type and number of attributes of the entity as well as its associations to other entities (the lexical similarity among attribute names is not used for clustering, as it could preclude synonyms; however, lexical similarity does help in manually labeling the clusters generated automatically). The two variables *number of attributes* and *number of foreign keys/links* taken together help to determine this type of evolution of similar concepts.

In addition to *number of attributes* and *number of foreign keys*, it would be good to be able to use type descriptions of attribute domains. However, it has not been possible to adequately capture type similarities due to the sheer number of entities and their descriptions when dealing with multiple heterogeneous systems.

In addition to static descriptions of objects reflected in the schema of a database, frequency of entity updates and usage profiles of users carry a cognitive load that reflects the dynamic properties of a database, and provides useful information that is exploited by the concept discovery process. The variables *number of users*, and *number of user types* help determine similarity in the user profile of different groups, as similar groups of users tend to use information pertaining to similar concepts to manage their business. The limitation of choosing the number of user types rather than actual user types makes this invariant to the composition of the group. However, within the clinical domain, the number of user types is from a small well-understood set. The variable *usage frequency* captures some dynamic properties, and gives a good indication of the use of each database object. The number of tuples associated with the entity and the number of join relationships in which the entity participates during query formulations are additional indicators for this purpose. The variables *number of tuples* and *number of foreign keys/links* help towards this, as its relationship to other objects often reveals more information about the object than the object itself.

The variable *number of tuples* taken by itself may not appear to be indicative, as it may be determined by the semantics of the application. However, similarity in the relative order of number of tuples within each system taken in conjunction with *usage frequency* is a function of usage of key concepts common to cooperating systems. To account for the fact that different systems may be in use for different periods of time, and to determine relative variances within each system, the variable *number of tuples* has been ranked within each system before it is used in the clustering process. Concept similarity is based on the relative order of ranking within each system. In the future, a reference to this variable means the ranked variable within each system.

The main endeavor here is to develop a *domain abstraction* to provide a conceptual view of heterogeneous information systems, and not to provide a *tightly integrated schema* of cooperating information systems.

ConceptDISH uses the principle of conceptual clustering to classify or group database objects into meaningful generic concepts common to the application domain.

The ConceptDISH Algorithm

A large number of clustering algorithms have been developed with different definitions of clusters and similarity among objects. Authoritative references on clustering techniques can be found in Hartigan (1975) and Everitt (1980). The ConceptDISH algorithm uses an optimization technique, as it is possible to have a fairly good idea of the possible number of clusters into which the entities of a CIS database (or any other participating information systems) can be grouped (from the results of empirical study phase described in Srinivasan & Ngu, 1995). To identify distinct concepts, a disjoint clustering algorithm is used, where observations are divided into clusters such that every observation belongs to only one cluster. Most clustering methods are biased toward finding clusters possessing certain characteristics related to size, shape, or dispersion. The approach used by ConceptDISH is to minimize within cluster variance, in preference to spherical clusters or clusters with equal number of members. Spherical clusters indicate equal distances between cluster means, which may not be the case when dealing with concepts embedded in legacy systems. Similarly forming clusters with equal numbers of members is also not desirable, as all concepts may not have identical representations in different legacy systems. Hence, choosing clusters with minimum within cluster variance is preferable for the problem on hand. For each cluster, the cluster summary contains a cluster number, the members of the cluster, maximum distance from the cluster seed to any observation, and the distance between the centroid and the current cluster and its nearest neighbor. Thus each cluster indicates a concept, and the members of the cluster indicate the entities that represent that concept cluster or belong to that concept group.

The ConceptDISH algorithm presented here uses the procedures ACECLUS and FASTCLUS (SAS Institute Inc., 1989), which, when used together, provide distinct concepts characterized by disjoint clusters with minimum within-cluster variance. This procedure uses a method called nearest centroid sorting. A set of points (observations) called cluster seeds are first automatically selected as a first guess of the means of the clusters. Each observation is assigned to the nearest seed to form temporary clusters. The seeds are then replaced by the means of temporary clusters formed and the process is repeated until no further changes occur in the clusters.

ConceptDISH Algorithm

```
   Let the maximum number of clusters be $N$
   Let the number of observations be $M$
   Let the number of variables measured for each observation
      be $S$
   Observation I lies in cluster P(I), [1 <= P(I) <= N]
   Let the cluster seed associated with cluster P(I) be L(I)
   Let $L_T$ be the set of cluster seeds at time T [1 <= $L_T$(I)
      <=N]
   Repeat for each cluster P(I)
   Repeat for each observation J, J ( $L_T$(I)
        compute D(J, $L_T$(I)) /* D(J, $L_T$(I)) is the Euclidean
        distance between Object J and cluster seed $L_T$(I) */
   end repeat /* for each observation */
   where D(J, $L_T$(I)) is minimum
   add J to cluster P(I)
   recompute $L_T$(I)/recompute cluster seed using means, $L_T$
   becomes $L_{T-1}$ /
end Repeat if $L_{T-1}$ = $L_T$ /* for each cluster P(I) */
```

The algorithm was tested using information from three existing clinical information systems in a leading hospital in Sydney. The actual experimental framework is described in Srinivasan (1997).

The results of the clustering experiment are discussed in the following section.

Experimental Results

Table 1 shows an illustrative subset of the database entities (objects) used in the experiment. For the purposes of clarity, entities of system 1 are shown in italics, while that of system 2 are shown in capitals, and the entities of system 3 shown in a normal font style. Each object is characterized by six variables. Table 2shows a subset of the actual data (objects and variables) used in the experiment. Objects with null values for any of the clustering variables are not included in the experiment, as a null value indicates that the object is not being used regularly. (This also accounts for the fact that all systems do not have representative members in all clusters.) Table 3 shows the results of applying the ConceptDISH algorithm on three clinical information systems used in a Sydney hospital.

TABLE 1. Entities used for concept clustering

System 1 (*Endocrinology*)	System 2 (COMMUNITY HEALTH)	System 3 (Allied Health)
Clinical-assessment	HCCASE	Clinical-area-code
Country	HCCONDITIONS	Clinical-stat-code
Diet-adv-group	HCCOUNTRY	Code-type
Diabetes-control	HCLANGUAGE	Compensible-status-c
Drug-dosage	HCLINK	Contact-type-code
Drug-treatment	HCPMI	Depr-activity
Doctor-details	NCPOSTCODE	Dept-act-code
Diet-treatment	HCPRINTERS	Dept-code
Hospital-admissions	HCPROBLEMS	Doctor-code
Language	HCPROCEDURES	Fin-class-code
Patient	HCSTAFF	Generic-code
Post-code	HCSTATISTICS	Grp-activity-code
Patient-diet-record	HCTEAM	Grp-cln-therapist
Patient-complication	HCUSER	Hosp-group
Pat-exercise	HCUSERDEFS	Hospital
Patient-history		Insurance-cover-code
Pat-treatment		Intervene-code
Review-bookings		Language-code
Type-of-treatment		Local-code
Visit		Marital-status-code
		Medi-diag-code
		Pat-activity
		Patient
		Pat-status-code
		Pat-ther-group
		Refd-out-for-code
		Refd-out-to-code
		Referral-source-code
		Referred-for-code
		Referral
		Referral-out
		Thera-disch-to-code
		Therapist
		Therapy-course
		Therapy-diag-code
		Therapy-grp-clinic
		Therapy-grp-code
		Ther-disch-reason
		Ward-id

A brief description of the three systems are given below.

System 1 is a clinical endocrinology system developed for managing chronic problems. This system is developed using POWERHOUSE, a fourth generation language. It holds data pertaining to patient visits, pathology results, diets, drugs prescribed, follow-up programs, and a variety of other clinical measurements.

System 2 is implemented as a hierarchical model using the RMS file system of DEC. This is mainly used by health

TABLE 2. Sample data set used for clustering

Entity-name	Usage frequency	Number of users	Number of user-types	Ranked tuples	Number of attributes	Number of links
Clinical-assment	20	5	2	6	5	3
Country	10	5	3	4	2	0
Diabetes-control	50	3	2	8	8	2
Diet-adv-group	10	5	2	0	2	1
Diet-treatment	20	5	2	6	9	3
Doctor-details	10	10	5	9	11	3
Drug-dosage	50	5	2	7	2	5
Drug-treatment	30	10	5	7	4	4
Hospital-admissions	5	5	2	3	3	2
Intervene-code	30	10	1	7	5	2
Language	10	5	3	5	2	0
Pat-exercise	50	5	2	8	8	5
Patient	50	10	5	8	80	18
Patient-complication	2	2	2	2	5	1
Patient-history	20	5	2	5	10	3
Patient-diet-record	30	10	3	6	5	5
Post-code	15	10	3	8	2	0
Review-bookings	20	10	2	6	6	3
Type-of-treatment	10	5	2	2	2	1
Visit	60	10	5	8	40	8

workers logging in from different geographical locations. The information stored includes client demographics, history, problem/diagnosis, treatment, and follow-up activities.

System 3 is a relational database system designed to meet the needs of different groups of allied health workers. The database holds data pertaining to both in-patients and outpatients. Patient data includes case referral, patient history, diagnosis and treatment plan, discharge plan, and follow-up activities.

The algorithm was trialed for a range of 8 to 12 clusters. The best clustering configuration was obtained for eight clusters. This range was chosen on the basis of background knowledge of the clinical application domain (Srinivasan, Ngu, & Gedeons, 1994). This is further confirmed by studies conducted by Stead and Hammond (1985), who have analyzed more than 300 electronic medical records. They have classified data stored in medical records into the following groups: (1) demographic data identifying socioeconomic data; (2) general nontime-dependent data such as tumor registry information and special patient needs; (3) problem or diagnosis and associated treatment; (4) time-oriented information such as physical findings, patient's complaints, disease progress; (5) data resulting from orders such as laboratory tests and X-rays; (6) therapies including diets, physiotherapy, etc.; and (7) encounter information such as admission data including dates, places, and care provider (Ramirez, Smith, & Peterson, 1994).

After the clusters are generated, they are labeled manually by the domain experts. Labeling at this point is not labor intensive; hence, scalable, as automatic concept clustering already reduces the information space of the underlying heterogeneous systems. Also, the concept set within the domain is well specified as the labels are drawn from the initial set of concepts identified by users during empirical studies (Srinivasan et al., 1994; Stead & Hammond, 1985) and further guided by the names of the cluster members (entities) themselves. Although clustering is done on the basis of a six-dimensional vector that characterizes each object, values of certain variables of member entities offer some insight into the nature of the clusters themselves.

Observations

(1) Cluster 1 consists mainly of code tables from all the three systems. Members of this cluster are characterized by low *number of foreign keys* (links), and low number of user types. This appears appropriate, as code tables usually have a higher level of security, are used by a smaller group of people, and are not linked to many other entities. Moreover, code table definitions often do not have many attributes in the clinical application domain.
(2) Cluster 2 has entities that indicate a medium level of usage due to medium range values for usage frequency. This cluster is characterized by a strong similarity in the number of association links. This cluster has been labelled as "therapy," mainly based on the semantics of the cluster members.
(3) Entities of cluster 3 seem to require a relatively larger *number of attributes* to describe adequately, and are characterized by fairly high ranked tuples. This cluster has been labeled as "progress."
(4) Cluster 4 has entities that deal with patient "encounters." Typically, these entities are accessed during each visit of the patient, are used in queries often, and are updated regularly. This is indicated by a combination of *usage frequency*, *ranked tuples*, and *number of attributes*.
(5) Cluster 5 has entities that represent "patient demographics." In the clinical application domain these entities are characterized by large *number of attributes*, fairly high *usage frequency*, as practically every user type needs this entity.

TABLE 3. Results of the clustering algorithm

Cluster number	Cluster name	Entity name	Usage frequency	Number of users	Number of user types	Ranked tuples	Number of attributes	Number of links
1	Code	*Country*	10	5	3	4	2	0
		Diet-adv-group	10	5	2	0	2	1
		Doctor-details	10	10	5	9	11	3
		Hospital-admissions	5	5	2	3	3	2
		Language	10	5	3	5	2	0
		Patient-complication	2	2	2	2	5	1
		Post-code	15	10	3	8	2	0
		Type-of-treatment	10	5	2	2	2	1
		HCCOUNTRY	10	5	3	3	2	0
		HCLANGUAGE	10	5	3	5	2	0
		HCPOSTCODE	50	25	5	6	2	0
		HCPROCEDURE	20	5	3	2	2	0
		HCSTAFF	20	20	5	5	11	1
		HCUPRINTERS	5	3	2	0	3	0
		HCUSER	5	1	1	1	6	1
		HCUSERDEFS	5	2	1	2	8	1
		Clinical-stat-code	20	10	1	2	5	1
		Code-type	20	10	1	2	3	0
		Dept-code	30	10	1	0	3	0
		Doctor-code	20	10	1	6	3	0
		Fin-class-code	20	10	1	3	3	0
		Generic-code	20	10	1	4	4	0
		Hospital	20	10	1	1	8	1
		Hospital	20	10	1	1	3	0
		Insurance-cover-code	20	10	1	1	3	0
1	Code	Language-code	20	10	1	3	3	0
		Local-code	20	10	1	9	6	2
		Marital-status-code	10	10	1	1	3	0
		Medi-diag-code	20	10	1	8	5	2
		Pat-status-code	10	10	1	0	3	0
		Therapy-diag-code	50	100	2	8	5	2
		Ward-ide	5	10	1	4	2	0
2	Therapy	*Drug-dosage*	50	5	2	7	2	5
		Pat-exercise	50	5	2	8	8	5
		Patient-diet-record	30	10	3	6	5	5
		Pat-ther-group	40	10	2	7	8	5
		Therapy-grp-clinic	40	10	1	6	7	5
3	Progress	*Diabetes-control*	50	3	2	8	8	2
		HCPROBLEMS	70	3	2	8	8	2
		Therapy-grp-code	80	5	2	8	16	4
4	Encounter	*Visit*	60	10	5	8	40	8
		HCCASE	120	25	5	9	25	3
		Grp-cln-therapist	120	50	2	9	32	3
		Pat-activity	80	50	2	8	11	4
		Referral	120	50	2	8	9	4
		Therapy-course	120	50	2	7	13	4
5	Demographics	*Patient*	50	10	5	8	80	18
		HCLINK	80	25	5	8	30	1
		HCPMI	80	25	5	7	36	9
		Patient	90	35	2	8	23	3
6	Nontime-dependent data	Refd-out-for-code	40	10	1	3	5	2
		Refd-out-to-code	40	10	1	4	5	2
		Referral-source-code	40	20	2	6	5	2
		Referred-for-code	40	10	2	5	5	2
		Thera-disch-to-code	40	10	1	2	5	2
		Ther-disch-reason	40	10	1	3	5	2
7	Diagnosis treatment	*Clinical-assment*	20	5	2	6	5	3
		Diet-treatment	20	5	2	6	9	3
		Drug-treatment	30	10	5	7	4	4
		Patient-history	20	5	2	5	10	3
		HCCONDITION	30	5	2	7	4	1
		HCTEAM	30	20	5	7	6	1
		Clinical-area-code	30	10	1	7	5	1

TABLE 3. continued

Cluster number	Cluster name	Entity name	Usage frequency	Number of users	Number of user types	Ranked tuples	Number of attributes	Number of links
8	General activities	*Intervene-code*	30	10	1	7	5	2
		Review-bookings	20	10	2	6	6	3
		HC-STATISTICS	50	20	3	8	7	3
		Dept-activity	50	50	2	9	7	3
		Dept-act-code	30	10	1	6	5	2
		Grp-activity-code	20	10	1	2	5	2
		Referral-out	40	40	2	5	9	4
		Therapist	40	40	2	5	1	7

The algorithm was trialed for a range of 8 to 12 clusters. The best clustering configuration was obtained for 8 clusters. This range was chosen on the basis of background knowledge of the clinical application domain (Srinivasan, Igu, & Geacon, 1994). This is further confirmed by studies conducted by Stead and Hamond (1985) who have analysed more than 300 electronic medical records. They have classified data stored in medical records into the following groups: 1) demographic data identifying socio-economic data; 2) general non-time-dependent data such as tumour registry information and special patient needs; 3) problem or diagnosis and associated treatment; 4) time-oriented information such as physical findings, patient's complaints, disease progress; 5) data resulting from orders such as laboratory tests and x-rays; 6) therapies including diets, physiotherapy, etc; and 7) encounter information such as admission data including dates, places, and care provider (Ramirez, Smith, & Peterson, 1994).

(6) Cluster 6 represents "nontime-dependent data" such as referrals. Entities in this cluster are characterized by medium values of *usage frequency* and *ranked-tuples.* On examination of just the numbers, this cluster has some similarity to cluster 2. However, this cluster has a lower number of links than cluster 2, and has a high level of similarity in the number of attributes. It is interesting to note that all members of this cluster are from system 3. A closer examination of system 3 showed that it actually caters to allied health departments that have a greater need for nontime-dependent data such as referrals.

(7) Cluster 7 represents the "diagnosis/treatment" concept, and is characterized by a similar number in *ranked tuples* as well as *usage frequency*. This cluster has higher value for *ranked tuples* compared to cluster 6.

(8) Cluster 8 is labeled as "general activities," indicating departmental or house-keeping activities. This cluster is characterized by similarity in *usage frequency* and *number of users*.

Although all member entities of a given cluster may not be completely indicative of the cluster type, this conceptual clustering approach reduces the overall information space, and provides an abstraction of the application domain. This conceptual binding brings together similar entities into common concept clusters that describe the generic features of the application domain. The concept clusters also serve common ontological model to promote cooperation among heterogeneous systems. The ConceptDISH approach thus advances the concept of intelligent integration of information by providing a means by which users and information systems can comprehend the data held in multiple local systems.

In summary, the ConceptDISH approach described in this section proposes a unique method to achieve heterogeneous database integration through a process of domain abstraction rather than traditional transformation of heterogeneous sources into a common model. The abstraction itself is based on discovering similar patterns in *metalevel information* available in systems designed with similar objectives. The strength of this approach lies in its flexibility to handle schema changes as a result of database evolution. As the discovery algorithm includes the usage pattern as one of the parameters, the discovered concepts can be refined with increase of usage. This approach is scalable, as the logging of information can be done in each cooperating system separately and before applying ConceptDISH.

Evaluation

To evaluate the quality of clusters generated by the clustering algorithm, we conducted two experiments. In the first experiment, the clusters were created and labelled manually by a domain expert choosing an appropriate name from the following eight concepts: (1) demographics, (2) problem, (3) therapy, (4) progress, (5) encounter, (6) orders, (7) general, and (8) codes. This is based on application domain knowledge gained through the empirical study and also studies by Stead and Hammond (1985). In the second experiment, the clusters were created automatically using ConceptDISH. In both cases, C4.5 (Quilan, 1991) was used as a tool for measuring inductive generalization. A separate training set (which is a subset of the data used) was initially used for supervised learning. C4.5 uses the paradigm of supervised learning, which requires that some predefined concepts be assigned to the objects being examined. After classification, the program produces an estimate of the error rates of misclassifying unseen cases using the examples provided. Table 4 summariZes the results of the two experiments. The results are expressed as a percentage that indicates the estimated error rates in inductive generalization.

The first experiment shows that purely on the basis of manual classification, it is difficult to obtain good inductive generalization using the available data. This is demonstrated by the large estimated error of 35%. As the volume and

TABLE 4. Estimated errors in inductive generalisation

Experiment	Error before pruning	Error after pruning	Estimated error for inductive generalisation
1	18%	18%	35.2%
2	6.7%	7.9%	18.5%

complexity of databases increase, it becomes very difficult for a domain expert to hold in balance all the factors for assessment and classification.

The second experiment shows that C4.5 can learn to reproduce the clusters found by the clustering algorithm, with a lower generalization error showing increased consistency and accuracy of the automatically generated clusters. C4.5 uses a crossvalidation technique that is a pessimistic estimate of error rates. This gives the error before pruning. Then C4.5 uses an additional heuristic that the probability of error rate cannot be determined exactly, but has itself a probability distribution that is summarized by a pair of confidence limits. Then C4.5 simply equates the predicted error rates at a leaf with this upper limit, on the argument that the tree has been constructed to minimize the observer error rate, thus giving the error rate after pruning.

Once discovered and labeled, these concepts form an initial ontology of the application domain, and help to provide customized external views to different user groups with their own preferred terminology.

The next section describes ConceptVIEW, the framework that uses the concepts discovered by ConceptDISH to provide an information retrieval facility that reengineers information from heterogeneous databases and provides a customized external presentation layer that is unique to each separate user type to reflect their particular contextual need.

Information Reengineering Framework: ConceptVIEW

The main objective of the ConceptVIEW framework is to allow each distinct type of user such as doctor, nurse, administrator, specialist, etc. (referred to as user type in the rest of the article) to access and view data from heterogeneous systems using familiar concepts. This framework allows users to browse and extend their search using concepts that are meaningful to them. In other words, each user type has the facility to retrieve information from heterogeneous information systems using their preferred vocabulary. User interaction is via a hypertext interface (Nielson, 1990), where the hypertext labels represent concepts meaningful to a user type. As per the I3 terminology (ISX Corporation, 1994), the ConceptVIEW framework provides "data and knowledge sharing service" and a "repository service." The components of the framework perform the role of a "facilitator" to provide the required external view to different user types. The major components that provide the required user orientation are described in the following sections.

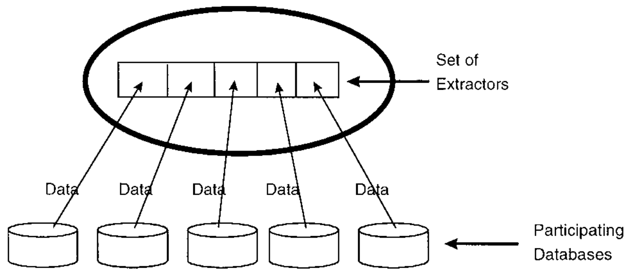

FIG. 2. Group data object.

Components of ConceptVIEW

Group Data Object

A Group data object (GDO) represents a customized concept for a particular user type. The set of GDOs with association links to other GDOs forms the starting point of information retrieval for that user type. Figure 2 shows the structure of a group data object.

Each GDO has a set of associated "extractors." Each extractor is capable of retrieving information from one local CIS. The extractors point to actual entities that are part of the cluster that represents the generic concept. The process of customizing the GDO involves selecting a suitable subset of extractors from each concept cluster. This customization function is performed by the GDO customizer. Developing a set of useful GDOs for a particular user type requires defining a user object for each user type.

User Object

The user object (UO) models the needs of a particular user type. The user object is pictorially represented in Figure 3.

The user object has two main components: (a) a set of GDOs that are useful to a user type represented by the user object; and (b) a Context Labeler that provides appropriate labels to the GDOs to support the interpretation differences among different user types. These labels represent concepts that are meaningful to a specific user type.

The user object modeler manages the user objects that are organized in the form of a tree structure. A generalization–specialization relationship exists between the different

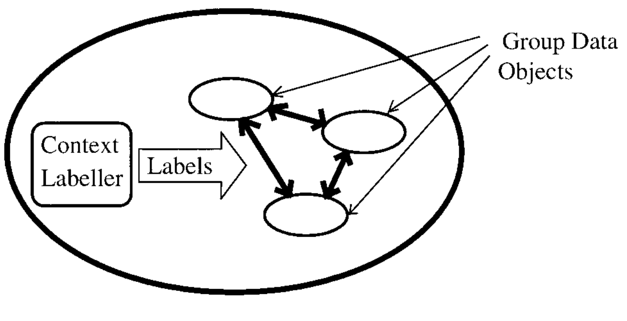

FIG. 3. User object.

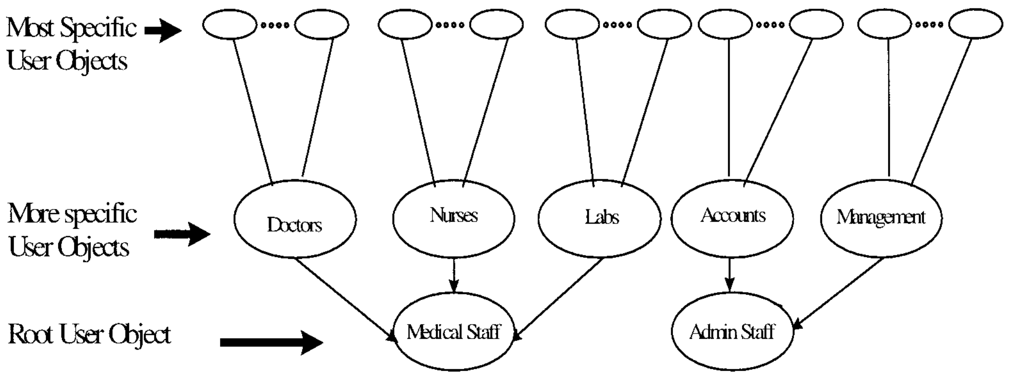

FIG. 4. User object hierachy.

levels of the hierarchical tree. Figure 4 shows the user object hierarchy. The root of the tree represents the UO for a broad base of user types with very distinct information needs. User objects that are more specific, are represented by the child UO. The lowest level of the object hierarchy can theoretically capture the contextual differences between different users in a particular user group. The child UO inherits all the specifications of the parent UO and has a few additional specifications. This hierarchical structure facilitates reuse of previously defined UOs in situations where new UOs are required.

Anchors

Anchors are links that control and coordinate the associations between GDOs. As defined in the standard hypertext reference model, the anchoring mechanism is necessary to support browsing within and across multiple structures. In the ConceptVIEW framework the anchors provide a browsing facilility that is essential to extend a query dynamically across multiple systems.

A repository in each UO stores a set of GDOs, anchors, and extractors that are suitable for that particular user type. Developing a set of useful GDOs for each user type requires an initial setup process that is outlined in the following section.

Initial Setup

The ConceptVIEW design strategy involves including the user to participate in the initial creation of a representational vocabulary. A user object dictionary is created with this representative vocabulary of the application domain. A subset of this vocabulary is set up for each user type to capture their preferred terminology. These preferred terms are used in the initial customization of a GDO for each UO. The choice of these terms is also influenced by common queries from local systems. Each GDO is then set up with extractors or pointers that point to the entities of the local CISs, which are part of the cluster representing a generic concept.

Table 5 shows how the same generic concept can be customized with different extractors for different user types. The first column represents the user type, the second column represents a generic concept meaningful to a user type shown in column 1. Column 3 shows the extractors associated with a particular concept for each user type. For the purposes of clarity and simplicity, only one concept and one extractor is shown here for each user type, although the system can handle multiple concepts and extractors. In reality, each user type would have a set of concepts customized as GDOs with appropriate extractors.

This table illustrates how the generic concept Treatment means different things to different user types. For the user type *Nurse*, it means *drug treatment*, for the user type *Dietitian* the concept also means *diet treatment*, while for the *Physiotherapist* it means *intervene code*.

ConceptVIEW Architecture

Figure 5 shows the ConceptVIEW architecture. The rectangular boxes show the main functions, the octagonal boxes show the resources used for information packaging, and the oval shapes show the components or objects used for implementation.

The functions of the various components are as follows:

(1) The UO dictionary is initially created as part of the initial setup process explained in the last section. The dictionary contains information about preferred terminology for each user type.
(2) The UO modeler uses the information from the dictionary to set up each UO in the UO hierarchy.
(3) The GDO customizer uses the concept clusters to generate a set of GDOs for each user type represented by a UO. These GDOs represent the concepts that are well

TABLE 5. GDO customisation

User-type	Generic concept cluster	Extractors
Nurse	Treatment	Drug-treatment
Dietitian	Treatment	Diet-treatment
Physio-therapist	Treatment	Intervene-code

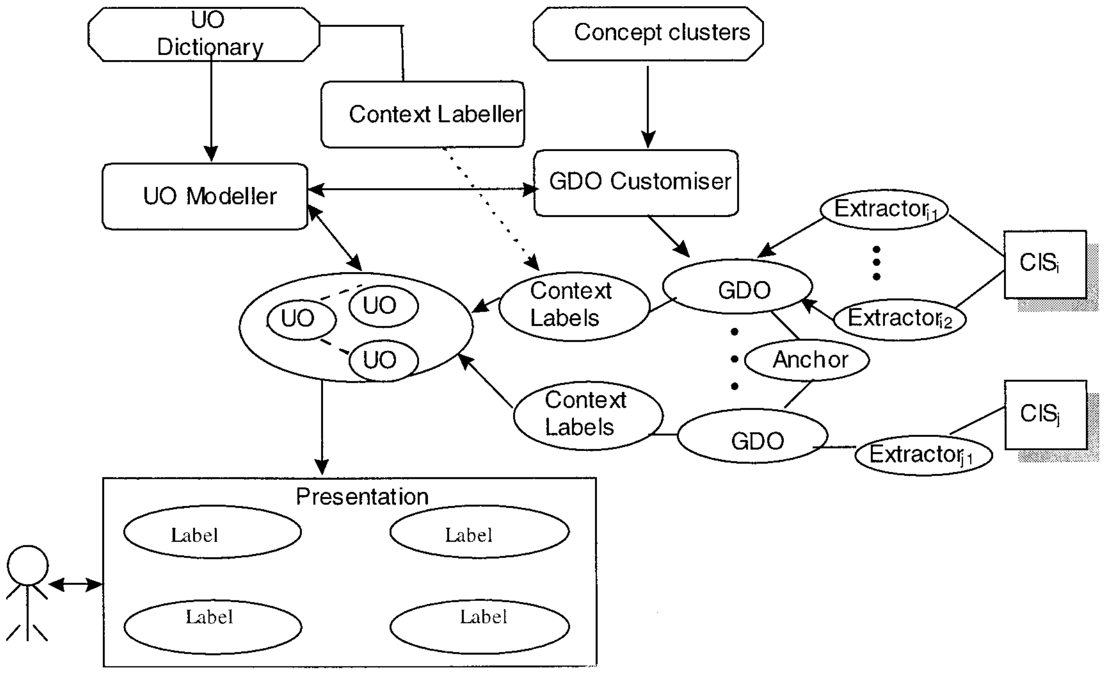

FIG. 5. ConceptVIEW architecture.

known to that user type in their application domain. A suitable set of extractors are also selected for these GDOs.

(4) A suitable anchor is used to associate each GDO with each other, and helps in dynamically extending the context of the query, depending of the user context.

(5) The context labeler maps the data retrieved from component CISs to the context of the user by providing appropriate labels for the retrieved data. The labels on the presentation layer represent customized concepts for a particular user type.

In a typical interaction, when a user logs in, the UO modeler determines the user type based on information in the UO dictionary, and passes on this information to the GDO customizer. The GDO customizer generates appropriate GDOs with appropriate extractors and anchors for the required UO. The Context labeller provides appropriate labels for each GDO. The presentation layer then displays the labels that represent the customized concepts (or GDOs) for that particular user type. When the user selects the required concept(s), the appropriate GDOs and the associated set of extractors are automatically invoked, and information is retrieved from the required CIS. Selecting additional GDOs invokes the anchors attached to the GDOs, thereby extending the query to include more concepts.

The customization process is explained with the help of some examples in the following section.

Examples

Three examples are illustrated to show how the information reengineering framework addresses the contextual differences among different user types in the clinical application domain. Each example relates to one of the contextual differences listed in an earlier section.

Example 1: Semantics

This example demonstrates the customization process that addresses the semantic differences between two user types; a *Nurse* user type and a *Dietitian* user type. Figure 6 illustrates a typical scenario when a *Nurse* user type interacts with the system. The labels displayed on the presentation layer correspond to the GDOs *Demographics, Progress, Treatment.* (This is a representative sample.) It is possible that other clinical users may have similar or overlapping display labels. The *Dietitian* user type may have the labels *Demographics*, *Treatment*, and an additional label *Encounter*. The figure on the right shows the presentation layer for a *Dietitian* user type. Upon selecting the same label *Treatment*, a different set of extractors are instantiated for the *Nurse* and the *Dietitian*, reflecting the differences in semantic understanding of the same term by the two user types.

Example 2: Abstraction

This example demonstrates the contextual difference that occurs between two user types who perceive varying levels of abstraction while dealing with the same concept. Figure 7 shows the presentation layers with some overlapping GDOs for the two user types *Nurse* and *Oncologist*. While treating a patient a *Nurse* mainly deals with “Drug treatment” details such as drug name, dosage, frequency. To the

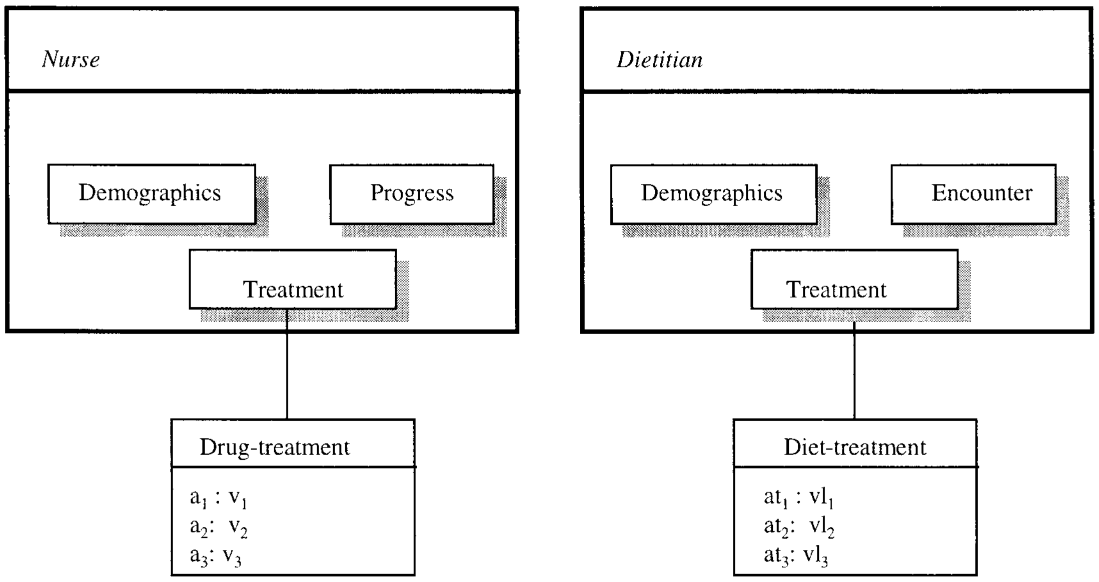

FIG. 6. User interaction—Example 1.

Oncologist, the label *Treatment* implies more detailed information about "Drug treatment" such as "potency," and other forms of "Therapy" such as chemotherapy and radiation therapy, etc.

Example 3: Association

This example illustrates the differences in association that contributes to the information retrieval strategy adopted by different user types while dealing with concepts that appear identical. At a high level they appear to use similar concepts. However, to reach a destination their traversal is determined by the information content at each node (GDO). This is illustrated by the traversal path used by the *Physiotherapist* and *Dietitian* user types. The Dietitian starts with *Treatment*, deals with "diet-treatment" of a patient, and then retrieves some information about a patient's "visit" details during a particular *Encounter*. The path traversed by the Dietitian is shown serially in the diagram. The Physiotherapist, on the other hand, deals with a completely different set of association links (as shown on the right side of Fig. 8) to retrieve infor-

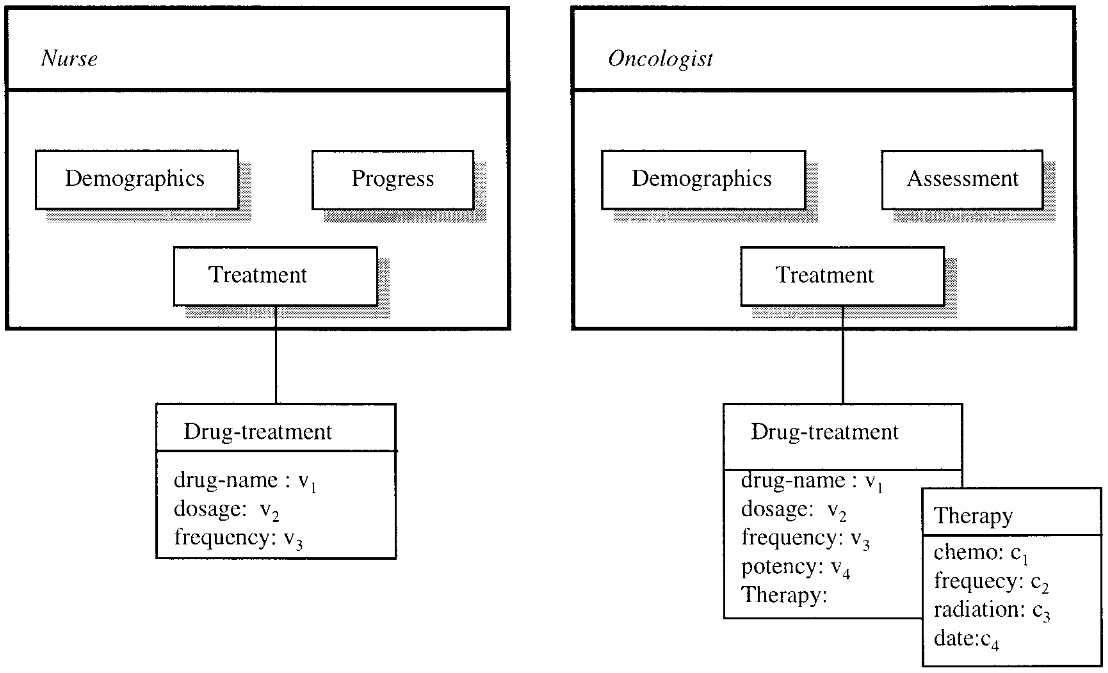

FIG. 7. User interaction—Example 2.

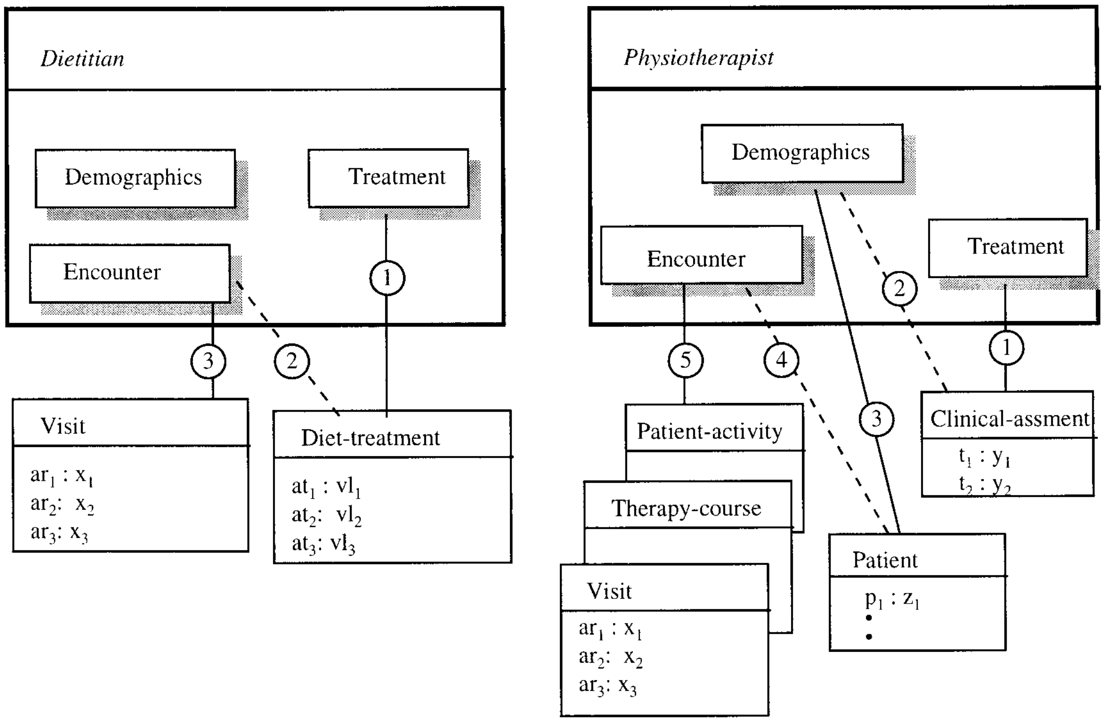

FIG. 8. User interaction—Example 3.

mation about a patient's "visit." The physiotherapist starts with a patient's *Treatment*, which involves "clinical assessment," of a particular patient whose details are retrieved using *Demographics*. The physiotherapist then wishes to see *Encounter* information that deals with "patient-activity" during the "therapy-course" that was carried out during the last "visit."

ConceptVIEW provides the required flexibility for each user type to extend a query in a context sensitive manner. This is achieved with the help of the anchors and extractors associated with each GDO in every user object.

The above examples show the importance and advantage of customizing information to suit the needs of each user type.

The ConceptVIEW framework described in this section provides an information packaging service that is customized to suit the needs of distinct user types in a given application domain. ConceptVIEW achieves this through the user object, group data object, and the context labeler. These three objects help to address the contextual differences among different user types. An initial setup process is required to initiate the customization process. The User Object Dictionary is initially created to store the terminology preferences of different user types, which is reflected through the labels in the presentation layer. In the current proposal the onus of choosing the appropriate terminology rests with the users. To improve the ConceptVIEW service, a promising extension would be to refine the initial setup by establishing a feed back loop from the presentation layer to the user object repository.

Related Work

There are a number of projects worldwide that are dealing with intelligent integration of heterogeneous sources. The approaches to integration vary. They can be organized in three logical categories, viz., structural, semantic, and intelligent integration. It is evident that researchers have approached the problem of heterogeneous database integration with a multitude of objectives—to develop a global schema that facilitates transparent access to distributed data, to integrate views, to facilitate interoperability without creating new objects, and to develop a new application with new objects that encompasses existing applications. Different goals of integration have produced different methodologies and techniques for integration.

Structural Integration

Batini et al. (1986) provide a comprehensive survey of different schema integration techniques. They define schema integration as the activity of integrating schemata of existing or proposed databases into a global, unified schema. The five steps to schema integration described are: preintegration, comparison, conformation, merging, and restructuring. These involve schema analysis, comparison, translation, and then integration. In heterogeneous systems where different types of models are used for developing the logical schema, comparison and analysis of schemata pose conflicts that are not readily resolved. However, it is difficult to compare schemata in the absence of a common basis for comparison. This neccesitates the translation of schemas to a common model as part of the preintegration phase. With a view to address this initial schema discrepancy problem, Sheth and Gala (1989) propose an integration approach that has four phases: preintegration, schema analysis, object integration, and schema restructuring.

A common underlying principle in structural integration is that they all use some form of common model (also called canonical model) to represent user views and perform some translation of local schemata to the canonical model. This need for a canonical model led to object-oriented models and extended relational models such as ERC+ being considered as candidates for the canonical data model (Saltor, Castellanous, & Garcia-Solaco, 1991).

Semantic Integration

While structural integration has been well defined in Sheth and Larson (1990) and Batini et al. (1986), such a comprehensive treatment from a semantic perspective has proven to be difficult. In an attempt to reconcile the semantic and schematic perspectives, Sheth and Kashyap (1992) present a semantic taxonomy to demonstrate semantic similarities between two objects and relate this to a structural taxonomy.

Various types of semantic relationships have been discussed in the literature. Many terms such as semantic equivalence, semantic relevance, semantic resemblance, semantic compatibility, semantic discrepancy, semantic reconciliation, and semantic relativism have been defined. There is no general agreement on all of these terms. Many authors have pointed out that semantic heterogeneity is the least understood, and hence, hardest to resolve in the context of heterogeneous databases.

There are a number of on-going research projects that address the semantic issues of database integration. Typical examples are FEMUS project at Swiss Institute of Technology and ETHZ (Yetongnon, Andersson, Dupont, & Spaccapietra, 1992) and Context technology Interchange Network (COIN) at MIT (Daruwala, Goh, Hofmeister, Hussein, Madnick, & Siegel, 1994). Although the two approaches are different, the common theme is to understand the semantics embedded in the applications. This requires knowledge of the entire application domain. This knowledge must be captured to facilitate some kind of intelligent translation between the source databases and the end users of the databases. This leads to the next section describing the knowledge-based approach to integration, also synonymously referred to as intelligent integration.

Intelligent Integration

Two major ideas or concepts are proposed in the literature to achieve intelligent integration. The first is based on establishing some form of intelligent cooperation between heterogeneous systems by transforming passive information systems into intelligent information processing *agents* (Brodie 1992; Papazoglous et al., 1992). Each type of information agent is characterized by its specific domain of expertise. For example, a wrapper agent translates schema information for each component information system, while a broker agent serves the function of matching the request with the potential service providers. Typical systems are SIMS (Arens, Knoblock, & Shen, 1996), RETSINA (Decker & Sycara 1997), and InfoSleuth (Bayardo et al., 1997).

The second approach in intelligent integration is based on the concept of *mediators* (Wiederhold, 1993) that provide intermediary services by linking data resources and application programs. The mediator provides integrated information without the need to integrate the individual data source. The TSMISS (Molina et al., 1997) project at Standford University is an important work in this area.

Both information agents and mediators require background knowledge of the application domain. The background knowledge is modeled in some sort of common vocabulary (ontology). This knowledge has to be "discovered"' from existing applications to provide the required intelligence needed for integration. In the InfoSleuth project, it is assume that the enterprise ontology model can be created by the domain expert.

Discussion

The ConceptDISH approach presented in this thesis focuses on discovering knowledge that is available in the form of database structures and usage patterns of legacy systems. Although most approaches deal with transformation and/or translation of data/metadata, this approach provides a domain abstraction, as it is based on concepts discovered from similar patterns in metalevel information in existing systems that were designed with similar objectives within a given application domain. We, thus, provide a methodology for helping users to set up the initial common ontology that forms the first step towards intelligent integration.

The ConceptVIEW provides an "intelligent" external layer that is customized for each user category, thereby enabling them to search for information from multiple systems in the context of their domain knowledge. The aim is to assist humans in information processing and integration, instead of automating integration which generally could not resolve semantics conflicts.

Another work that has a certain similarity to the conceptual framework presented here is the architecture proposed by Milliner, Bouguetaya, and Papazoglou (1995). They propose a scheme where existing information systems are treated as database nodes that are dynamically clustered around current areas of interest, also referred to as global concepts. Database nodes then form links to each global concept and associate an updatable "weight" with each global concept. When several database nodes all link strongly to the same concept with weight 10/10, a dynamic cluster of database nodes is formed. The main difference between the architecture in Milliner et al. (1995) and the framework proposed in this article, is that they talk of clustering "database nodes" into common interest areas, while this article discusses clustering “entities” of multiple databases into generic concepts based on similarity in meta-level information available in existing databases. At a philosophical level, in the approach proposed by Milliner et al. (1995), the participants contribute to the global framework, whereas in the ConceptDISH approach proposed in this article, the participants are the very cause of the conceptual framework.

Conclusions

The approach to integration presented in this article is based on establishing cooperation among existing legacy systems at a conceptual level. This conceptual level serves as a domain abstraction, and reflects the concepts that are common to information systems within the same application domain that need to cooperate with each other. The underlying philosophy is that similarity in metalevel information occurs because users with similar background knowledge and common information needs will specify and use information systems in similar ways. The similarity in background knowledge is reflected in comparable schema descriptions, and the similarity in information needs are reflected in highly related querying and usage patterns. This metalevel information, therefore, includes both static and dynamic properties of information systems. The significance of this approach using metadata mining is that it overcomes the problems associated with raw data, which is often incomplete, and may have limited breadth and coverage. Although the most integration approaches propose a common query language to retrieve data from heterogeneous systems, the framework proposed here reengineers information such that individual user groups have the facility to browse and interpret information from multiple systems within their own context using familiar concepts. Instead of automating the difficult semantic integration process, we provide a framework to assist users to perform semantic integration.

It is important to bear in mind that the concept clusters provide a domain abstraction of the application domain, and offer a basis for conceptual integration. They do not represent complete information of the entire application domain. An ideal condition would be to perform usage audits at regular intervals, and feed it to the discovery algorithm to continuously improve the quality of clusters. One of the limitations in this approach is that there is still a fair amount of manual preprocessing required before the concept discovery algorithm can be applied. Even after the discovery there is a manual labeling process that has to be performed to make the discovered concepts useful and meaningful.

The approach used by ConceptDISH and ConceptVIEW is flexible and dynamic in that the conceptual integration process can be a frequent activity. As usage patterns are utilized to discover concepts, increased usage can help to refine the concepts further. The information reengineering approach presented here addresses the uniqueness of each user group and allows contextual interpretation of information using terminology that is initiated and preferred by different user groups. This strategy allows users to query and retrieve information at a conceptual level determined by the user seeking information.

Developing a metadata resource directory that can support intelligent retrieval of information from the dynamic and diverse WWW resources is becoming more important, as information that is available on the Internet is growing exponentially. The work presented in this article could be carried forward to provide a tool set to discover and customize concepts from internet resources. For example, we could have different “discovery agents” for different types of sources, “mediators or customizing agents” that could dynamically transform and package the information for different user groups. Eventually these tools could help in developing a metadata resource directory that could be used to provide a new type of service in the internet.

Acknowledgment

This work was performed while the first and the third authors were affiliated to the University of New South Wales. The authors also wish to acknowledge the support provided by South Eastern Sydney Area Health Service for making valuable data and resources available for performing the experiments.

References

Ahmed, R., Smedt, P.D., Du, W., Kent, W., Ketabchi, M.A., Rafii, A., &. Shan, M. (1991). The PEGASUS heterogeneous multidatabase system. IEEE Computer, 24(2), 19–27.

Arens, Y., Knoblock, C.A., & Shen, W. (1996). Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, 6(2), 99–130.

Batini, C., Lenzerini, M., & Navathe, S.B. (1986). A comparative analysis of methodologies for schema integration. Computing Surveys, 18(4).

Bayardo, R., et al. (1997). InfoSleuth: Agent-based semantic integration of information in open and dynamic environments, in SIGMOD RECORD. Proceedings of the ACM SIGMOD (195–206). Tuson, AZ: ACM Press.

Brodie, M. (1992). The promise of distributed computing and the challenges of legacy information systems. In Proceedings of IFIP DS-5 semantics of interoperable databases systems. Lorne, Australia: Chapman & Hall.

Collet, C., Huhns, M.N., & Shen, W. (1991). Resource Integration using a large Knowledge Base in Carnot. IEEE Computer, 24(12), 55–62.

Daruwala, A., Goh, C., Hofmeister, S., Hussein, K., Madnick, S., & Siegel, M. (1994). The context Interchane Network prototype. Proc. IFIP DS-6 Semantics of Interoperable Systems. Atlanta, GA.

Decker, K., & Sycara, K.P. (1997). Intelligent adaptive information agents. Journal Intelligent Information Systems, 9(3), 239–260.

Everitt, B. (1980). Cluster analysis (2nd ed.), New York: Halsted Press, Division of John Wiley & Sons.

Hartigan, J.A. (1975). Clustering algorithms. New York: John Wiley & Sons.

Hsiao, D.K., Neuhold, E., & Sacks-Davis, R. (Eds.). (1992). Proc. IFIP DS-5 Semantics of interoperable database systems. Lorne, Australia: Chapman & Hall.

ISX Corporation. (1994). Intelligent integration of information (I3). [online] http://isx.com/pub/I3. Author.

ISX Corporation. (1995). Draft 8, appendix B: I3 glossary. Reference architecture for I3. Author.

Kim, W., & Seo, J. (1991). Classifying schematic and data heterogeneity in multidatabases systems. IEEE Computer, 24(12), 12–18.

Krishnamurthy, R., Litwin, W., & Kent, W. (1991). Language features for interoperability of databases with schematic differences. In Proc. ACM-SIGMOD (pp. 40–49). New York: ACM Press.

Litwin, W., & Abdellatif, A. (1986). Multidatabase interoperability. IEEE Computer, 19(12), 10–18.

Litwin, W. (1989). MSQL, a multidatabase language. New York: Elsevier Science Publishing.

Meersman, R. (Ed.). (1997). Database application semantics. London: Chapman and Hall.

Milliner, S., Bouguettaya, A., & Papazoglou, M. (1995). A scalable architecture for autonomous heterogeneous database interactions. In Proceedings of the 21st VLDB Conference (pp. 515–526).

Molina, G., et al. (1997). The TSIMMIS approach to mediation: Data models and languages. Journal of Intelligent Information System, 8(2).

Nielson, J. (1990). Hypertext and hypermedia. New York: Academic Press.

Ouksel, A., & Sheth, A. (Eds.). (1999). SIGMOD record. Semantic Interoperability in Global Information Systems, 28(1). ACM Press.

Papazoglou, M.P., Laufmann, S.C., & Sellis, T.K. (1992). An organizational framework for cooperating intelligent information systems. International Journal of Intelligent and Cooperative Information Systems, 1(1), 169–202.

Piatetsky-Shapiro, G., & Frawley, W.J. (1991). Knowledge discovery in databases. AAAI Press/MIT Press.

Quinlan, J.R. (1991). C4.5 programs for machine learning. San Mateo, CA: Morgan Kaufmann.

Ramirez, J.C.G., Smith, L.A., & Peterson, L.L. (1994). Medical information systems, characterisation and challenges. SIGMOD Record, 23(3).

Saltor, F., Castellanos, M.G., & Garcia-Solaco, M. (1991). Suitability of data models as canonical models for federated dataabses. SIGMOD Record, 20(4), ACM Press.

Saltor, F., Castellanos, M.G., & Garcia-Solaco, M. (1992). Overcoming schematic discrepancies in interoperable databases. In Proceedings of IFIP DS-5 semantics of interoperable database systems (pp. 272–301).

SAS Institute Inc. (1989). SAS/STAT® user's guide (version 6, 4th ed., volume 1). Cary, NC: SAS Institute Inc.

Segev, A., & Sheth, A. (Eds.), (1991). SIGMOD Record, 20. ACM Press.

Wiederhold, G. (1993). Intelligent integration of information. Proceedings of the ACM SIGMOD, 22(2), 343–437.

Sheth, A., & Larson, J. (1990). Federated database systems for managing distributed, heterogeneous and autonomous databases. ACM Computing Surveys, 22(3).

Sheth, A., & Kashyap, V. (1992), So far (schematically), yet so near (semantically). In Proc. IFIP DS-5 Semantics of Interoperable Database Systems. Lorne, Australia: Chapman & Hall.

Sheth, A. (1991). Semantic issues in multidatabase systems. SIGMOD Record, 20(4). ACM Press.

Siegel, M., & Madnick, S.E.. (1991). A meatadata approach to resolving semantic conflicts. In Proceedings of the 17th international conference on very large databases (pp.133–145).

Srinivsan, U., & Ngu, A.H.H. (1995). Information reengineering for cooperative clinical information systems. In Proceedings of applications of databases conference ADB95, Santa Clara.

Srinivasan, U., Ngu, A.H.H., & Gedeon, T. (1994). Discovey of generic concepts from heterogeneous clinical information systems. In Proceedings of second Singapore international conference on intelligent systems.

Srinivsan, U., Ngu, A.H.H., & Gedeon, T.D. (1997). Concept Clustering for cooperation. In Proceedings of the IFIP DS-6 Conference, Meersman, E. (Ed.), Database application semantics. London: Chapman and Hall.

Srinivasan, U. (1997). A framework for conceptual integration of heterogeneous databases. PhD Thesis, University of New South Wales.

Stead, W.W., & Hammond, W.E. (1985). Computer based medical records, the centrepiece of TMR. M D Computing, 5(5).

Wiederhold, G. (1996). Value-added mediation in large-scale information systems, presented at IFIP DS-6 conference, Atlanta, May 1995. In Meersman, E. (Ed.), Database application semantics. London: Chapman and Hall.

Yetongnon, K., Andersson, M., Dupont, Y., & Spaccapietra, S. (1992). Building a federated database: The FEMUS experience. Proc. of IFIP DS-% semantics of interoperable databases systems. Lorne, Australia: Chapman & Hall.