Automated Resolution of Semantic Heterogeneity in Multidatabases

M. W. BRIGHT
IBM Federal Systems Company

A. R. HURSON and S. PAKZAD
Pennsylvania State University

A multidatabase system provides integrated access to heterogeneous, autonomous local databases in a distributed system. An important problem in current multidatabase systems is identification of semantically similar data in different local databases. The Summary Schemas Model (SSM) is proposed as an extension to multidatabase systems to aid in semantic identification. The SSM uses a global data structure to abstract the information available in a multidatabase system. This abstracted form allows users to use their own terms (imprecise queries) when accessing data rather than being forced to use system-specified terms. The system uses the global data structure to match the user's terms to the semantically closest available system terms. A simulation of the SSM is presented to compare imprecise-query processing with corresponding query-processing costs in a standard multidatabase system. The costs and benefits of the SSM are discussed, and future research directions are presented.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed databases*; H.2.4 [**Database Management**]: Systems—*distributed systems*; H.2.5 [**Database Management**]: Heterogeneous Systems

General Terms: Design

Additional Key Words and Phrases: Federated database, imprecise queries, multidatabase, schemas, semantic heterogeneity, simulation

1. INTRODUCTION

Computer applications in general, and databases in particular, are an integral part of the daily function of different groups of users and organizations. Databases in each of these environments have developed independently to meet specific requirements. Moreover, different database management systems (DBMSs), which are usually incompatible with each other, have evolved

Authors' addresses: M. W. Bright, IBM Federal Systems Company, 800 North Frederick Avenue, Gaithersburg, MD 20879; email: mikebright@vnet.ibm.com; A. R. Hurson and S. Pakzad, Department of Computer Science and Engineering, The Pennsylvana State University, University Park, PA 16802; email: A2H@ecl.psu.edu and Pakzad@ecl.psu.edu

to meet the varying needs in these independent environments. However, in today's networked world, separate autonomous data sources, "islands of information" [Andrew 1987], are no longer able to meet increasingly sophisticated user needs. Related information important to a global application or request may exist in multiple, incompatible local databases. Users cannot be expected to manage system details of sending multiple requests in different languages (and possibly different data models) to multiple information sources. Multidatabase systems provide integrated global access to autonomous, heterogeneous local databases with a single, relatively simple, request [Bright et al. 1992; Hurson and Bright 1991b; Litwin 1988; Litwin and Zeroual 1988; Sheth and Larson 1990; Thomas et al. 1990].

1.1 Semantic Identification and the Summary Schemas Model

An important consequence of the local autonomy and heterogeneity of multidatabase systems is that semantically similar pieces of information may have very different names and different data structures in separate local databases. Local data access terms are developed to meet specific local requirements and will not be globally consistent. Multidatabase designers have created methods to integrate semantically similar, but syntactically different, data entities. However, these methods all assume that database designers or users can identify semantically similar entities despite the representation and naming differences. Without intimate knowledge of the structure of all local databases, this assumption is invalid. The Summary Schemas Model (SSM) has been developed as an extension to multidatabase systems to provide linguistic support to automatically identify semantically similar entities with different access terms [Bright and Hurson 1991; Hurson and Bright 1991a]. A database schema is a group of access terms (database name, relation names, and attribute names) which describe the structure and content of the data available in the database. A summary schema is a concise, more abstract description of the semantic contents of a group of input schemas. The SSM uses specific linguistic relationships between schema terms to build a hierarchical global data structure which describes the information available in all local databases in an increasingly abstract form. The data structure relates local-access terms which are semantically similar. The SSM provides intelligent, user-friendly access to multidatabase systems. In existing multidatabase systems, global integration is typically a very labor-intensive task, while SSM processing is largely automated. Many multidatabase systems create large global schemas which are difficult to create, maintain, and store. The SSM global data structure is much smaller and is easier to create, maintain, and store.

The SSM allows multidatabase users to submit global database queries that described requested data in terms that are meaningful to each user. Consequently, users are not required to employ existing system access terms —the result is an imprecise query. Imprecise means the user may accept a range of system access terms as valid representations of his/her intent. The

SSM uses the global data structure and online linguistic tools to interpret the user's imprecise query and associate it with the precise local system access terms that are semantically closest to the user's terms. Hence, it allows users to describe their semantic intent while the system provides the best possible response. The ability to submit imprecise requests is important to making global data access user friendly and is also useful to global-database designers. Processing imprecise user queries requires some overhead relative to precise multidatabase query processing. However, this overhead is small relative to overall query-processing costs; moreover, the user function provided is not available in existing systems.

1.2 Structure of the Article

This article presents the Summary Schemas Model as an enhancement to existing multidatabase systems. Sections 2 and 3 review background material in multidatabase systems and in linguistic research, respectively. Then, the architecture and use of the SSM is presented in Section 4. Query-processing techniques are explored, with particular emphasis on resolving imprecise user references. Section 5 presents simulation results which show the effectiveness of the SSM. The overhead cost of imprecise-query processing compared to standard multidatabase language system (precise) query processing is quantified. Finally, Section 6 summarizes the article and discusses future extensions to the SSM.

2. MULTIDATABASE SYSTEMS

A multidatabase is a distributed system that acts as a front end to multiple local DBMSs or is structured as a global system layer on top of the local DBMSs [Breitbart et al. 1992; Bright et al. 1992; Sheth and Larson 1990; Staniszkis 1986]. Multidatabase are also referred to as federated databases or heterogeneous distributed databases. Local DBMSs may be heterogeneous (different data models and/or implementations of the same data model) and may exist prior to joining the multidatabase. Although the local node must maintain some global function in order to interface with the global system, the local DBMS participates in the multidatabase without modification. The local DBMS retains full control over local data and processing (local autonomy). Cooperating with the global system and servicing global requests are strictly voluntary. The global system provides a means (a global schema or multidatabase language features, Sections 2.1 and 2.2) of resolving the differences in data representation and function between local DBMSs. This resolution capability is necessary because the same information may be maintained at multiple locations in differing forms. Different representations of semantically similar data may include different naming conventions, formats, data structures, levels of abstraction, and/or completeness/consistency of the data. A number of multidatabase projects have been reported in the literature [Bright et al. 1992; Litwin 1988]. Examples of multidatabase projects include ADDS [Breitbart and Tieman 1984], Calida [Jakobson et al.

1988], DQS [Belcastro et al. 1988], EDDS [Bell et al. 1987], Mermaid [Templeton et al. 1987], MRDSM [Litwin 1985b], Pegasus [Raffii et al. 1991], and PRECI* [Deen et al. 1987]. The two major design approaches for multidatabase systems are explored in the following sections.

2.1 Global-Schema Multidatabases

The global-schema approach to multidatabases is a direct outgrowth of distributed databases [Ceri et al. 1987]. The global schema is just another layer, above the local external schemas, that provides additional data independence. Global users essentially see a single, large, integrated database. A major difference, however, is the lack of global control over local decisions. The global schema is usually replicated at each system node. Most global-schema multidatabases use a relational data model and some variant of SQL for the global-access language. However, object-oriented data models and object-oriented access languages are becoming more popular.

Global-schema design takes the independently developed local schemas, resolves semantic and syntactic differences between them, and creates an integrated summary of all the information from the union of the local schemas. Global-schema design is also referred to as view integration [Batini et al. 1983]. This process is much more difficult than just taking a union of the input schemas for several reasons. The most important reason (for our purposes) is that information about the same real-world object may occur in multiple local databases and have completely different representations. Despite the methodologies, algorithms, and heuristics which have been defined to help automate parts of the schema integration process, this process is still very human labor intensive [Batini et al. 1986]. Global-Database Administrators (DBAs) are required to design the global schema. These designers must have extensive knowledge of all the input schemas and the user requirements of the global system to decide how to integrate the inputs. Each of the local schemas is assumed to be optimized to local requirements. The global DBA must understand all the local optimizations and consider them when trying to create efficient global structures. The amount of global knowledge required, about what is being integrated and how to integrate it, is a major problem with the global-schema approach.

A global schema can be a very large data object. The sheer size can make it a problem to replicate at nodes with limited storage facilities. The popularity of personal computers and small DMBSs which may want to join the multidatabase system makes this an important problem. Some systems get around this problem by only replicating the global schema at specified server nodes.

Global DBAs must also maintain the global schema in the face of arbitrary (since the local DBMSs are autonomous) changes to local schemas. The literature is largely silent on how this is done. Changes to local schemas, including addition and deletion of whole nodes, must be reflected by corresponding changes in the global schema. The integration techniques used in global-schema design and the changes in local data representations at the global level can make the mapping of changes to the global schema a complex

problem. Local changes may force the DBA to reconsider may design decisions made during the initial integration process—with wide-reaching consequences.

2.2 Multidatabase Language Systems

The multidatabase language approach is an attempt to resolve some of the problems associated with global schemas, such as up-front knowledge required of DBAs, extensive development time to create the global schema, large maintenance requirements, and processing/storage requirements placed on local nodes. A multidatabase language system puts most of the integration responsibility on the user, but alleviates the problem by giving the user many functions to ease the task and providing a great deal of control over the information. Most multidatabae languages are relational, similar to SQL in the standard capabilities, but extend the function significantly. Litwin and his colleagues have argued persuasively for multidatabase languages and performed much research in this area [Fankhouser et al. 1988; Litwin 1984; 1988; Litwin and Abdellatif 1986; 1987].

Most of the language extensions beyond standard database capabilities are involved with manipulating differing data representations. Since representation differences exist when the user submits a query, the language must have the ability to transform source information into the integrated representations most useful to the user. It is particularly desirable in this context to make the language nonprocedural. The multidatabase system should be capable of making good implicit decisions in interpreting what the user wants to accomplish and providing many functions by default. The more complexity the system can automatically handle, the easier the system will be to use. Examples are the ability to iterate operations over multiple, slightly varying objects [Litwin 1984] and the ability to do implicit joins [Litwin 1985a]. In summary, the multidatabase language approach shifts the burden of integration from global DBAs (the global-schema approach) to users and local DBAs. Multidatabase language systems trade a level of data independence (the global schema hides duplication, heterogeneity, and location information) for a more dynamic system and greater control over system information.

3. LARGE-SYSTEM INTERFACES AND LINGUISTIC RESEARCH

The Summary Schemas Model (SSM) draws heavily from previous work in large-system user interfaces and in linguistic theory. The large amount of information available in a multidatabase system presents difficult challenges to user interface designers. The system must provide aids which guide the end user efficiently to the desired information. Identifying the semantic relationship between terms using linguistic theory is an important building block for the SSM. Recent advances in online dictionaries and thesauruses make it possible to apply linguistic theory in an automated fashion. Previous work on handling imprecise data values and on defining the semantic simi-

larity between terms has been extended to allow users to submit imprecise queries to the SSM.

3.1 Large-System User Interfaces

In a small data access system, it is reasonable to expect all users to learn the exact names for different entities and use the system-designated access terms to manipulate information. In this context, an access term is a descriptive identifier for a data entity—specifically, relation names and attribute names in the relational data model. As systems grow in size, there are more data entities and more corresponding access terms. In the large distributed systems represented by the multidatabase environment, it is unreasonable to expect users to know the exact system access terms for more than a narrow segment of the information. Not only is there a potentially huge number of access terms, but the naming conventions, data structures, and user paradigms for different local databases can vary significantly.

Another aspect of the size of the multidatabase environment is that the system has more applications and more potential for new applications to be added (relative to a single system). Therefore, it is likely that there will be many new and infrequent system users. It is possible that these users will have a different mental model of the data being accessed than the world model encoded in the global database [D'Atri and Tarantino 1989]. These differing views of the data to be accessed will hamper the user's attempts to manipulate the proper data. Because the naming and syntax of user requests may differ from the naming and syntax of system data, the user interface should have some intelligence to interpret the semantics of the user request.

3.2 User Interface Techniques

Three techniques—browsing, connection under logical implication, and generalization—have been proposed to aid users in searching and understanding the data represented in a system [D'Atri and Tarantino 1989]. Generalization has proven difficult for large and complex systems. Connection under logical implication places most of the burden on the system, but seems to be the most promising technique for large systems.

The problem with allowing a user to specify data in his/her own terms—i.e., connection under logical implication—is that different users are very likely to use different access terms for the same entity. One study showed that the probability of two subjects picking the same term for a given entity ranged from 7% to 18% [Furnas et al. 1987]. The study looked at several design methods aimed at increasing the probability of matching access terms for equivalent entities and concluded that the best matching performance would be obtained with a system that allowed unlimited synonyms for each entity in the system. In other words, the user would specify data using his/her own terms, and the system would match the user's terms to the closest synonymous term in the actual system data. Methods that rely on the system interpreting the user's request (connection under logical implication and

matching synonyms) have the problem that a user's request may be vague or imprecise. Several interfaces based on the concepts of dialogue and semantic distance have been developed to deal with imprecision in user requests [Motro 1989].

The proposed Summary Schemas Model provides a user-friendly interface by allowing users to specify queries in their own terms and/or to use imprecise terms for data references. The summary schemas and the online taxonomy are used to map these terms to the semantically closest access terms that actually exist in the system. Therefore, the system is very tolerant for new and infrequent system users. The summary schemas are a global data representation that can be browsed at various levels of abstraction and with varying amounts of detail shown.

3.3 Related Linguistic Research

3.3.1 *Semantic Relationships.* One of the motivating concepts for the SSM is that the words one uses to express ideas can be related in a unifying taxonomy based on the semantic content of the words. In his thesaurus, one of Roget's goals was to order a collection of words based on their meaning, rather than their (arbitrary) spelling [Dutch 1965]. Strong and Drott [1986] also proposed a classification based on semantic categories. The SSM makes extensive use of two particular semantic relationships between terms, *synonymy* and *hypernymy*. Strong synonyms are semantically equivalent terms that can be substituted for each other in all contexts without any change in meaning. Weak synonyms are semantically similar terms that can be substituted for each other in some contexts with minimal change in meaning. The hypernym of a word is a term with a broader, more general meaning (a genus term). The opposite relationship, a more specific meaning, is called hyponymy. There are many different possible hypernym relationships between two terms [Klavans et al. 1991b]. Examples are A "is-a" B, A is "part-of" B, A is a "member-of" B, A is a "form-of" B, etc. Also, hypernymy between verbs is slightly different than noun hypernymy [Collier and Fellbaum 1988]. The SSM currently uses weak synonymy and a generic, inclusive form of hypernymy. Other lexical relationships between terms can be defined [Aitchison and Gilchrist 1987], but are not currently used by the SSM.

When grouping terms semantically, it is important that each term have a precise meaning. However, many English terms have multiple definitions. Homographs are totally different words with the same spelling, while polysemy is the fact that one word can have multiple senses. An example of homographs is "bow," meaning the object used to shoot arrows, and "bow," meaning to bend at the waist. An example of polysemy is the word "room," which can mean space, a space within a building enclosed by walls, living quarters, or the people gathered together in a room. The Lexical Systems group at IBM's Watson Research Center has done extensive work on automated sense disambiguation [Chodorow et al. 1988; Klavans et al. 1991a; Ravin 1988; 1989]. They have created tools that use the linguistic information

in online dictionaries and thesauruses to disambiguate term senses and the semantic links between them. A consequence of the idea that words can be ordered according to their meaning is that a quantitative measure can be defined that specifies the semantic distance between words [Chodrow et al. 1988; Ravin 1988]. Another tool from the Lexical Systems group can find the path between two terms following specified semantic links. Hence, the semantic distance between the terms can be quantified by measuring the path length.

Work at the University of Illinois (at Chicago) uses the idea of common concepts (called facets in Aitchison and Gilchrist [1987]) to define semantic relationships between attribute names [Yu et al. 1991]. A concept is defined as a specific characteristic (or attribute) of a term, and a term is defined using multiple concepts. For example, an airplane is a mechanical device, but it also has the concepts of transportation, seating capacity, and air travel. The University of Illinois model builds hierarchies of concepts by hand. Then the system can automatically compare attribute terms for semantic similarity by calculating the intersection of their concept sets. This system of multiple concepts per term is more sophisticated than Roget's single hypernym approach, but the concept hierarchies must be constructed manually for each problem domain.

3.3.2 *Information Retrieval Concepts.* Traditional work in information retrieval systems has concentrated on bibliographic or unformatted databases [Aitchison and Gilchrist 1987; Lancaster 1986]. Users submit requests which contain access terms (from a controlled vocabulary) they are interested in, and the information retrieval system matches the user access terms with archived documents that have corresponding access terms. This concept is used in the SSM by theoretically considering local database schemas as documents. The terms in a schema are the access terms used by the system to index that schema (document). User queries are parsed for access terms which are matched with appropriate access terms from available schemas.

3.3.3 *Summary Schema Model Requirements.* The SSM requires a general taxonomy with synonym links, hypernym links, and word sense definitions. However, much of the work in the thesaurus field has concentrated on specialized thesauruses for limited fields or with restricted vocabularies, rather than generalized thesauruses for an entire language [Aitchison and Gilchrist 1987]. A specialized vocabulary can be more precise for classifiers and system-level users, but it requires extensive knowledge and effort to create. Since a multidatabase can contain information from many fields and many of the purposes of the SSM are user oriented, a broad, general vocabulary is required. SSM users are allowed to submit global queries using their own terms to describe data, so an entry vocabulary equivalent to the contents of a collegiate-level dictionary is appropriate. Note that the SSM only uses relationships between words—it does not attempt to understand the deep meaning of terms. The SSM is intended as a user aid, rather than a knowledge base such as the Cyc work at MCC.

3.4 Imprecision

Identifying specific information in a large system with an uncontrolled vocabulary (for schema terms and/or data values) is a difficult task. Often the system will contain relevant information that does not precisely match a user's request. Imprecision (or vagueness) can result from a mismatch between the user's and designer's world views or from the fact that local databases do not precisely represent reality. An example of the latter is a database that records the temperature as "cold" rather than the precise numeric value for degrees. When dealing with imprecise information, a system must quantify the similarity between data values and/or access terms.

3.4.1 *Imprecise Data Values.* Previous work on imprecision in databases has concentrated on imprecision in data values. The literature has addressed several directions for handling imprecise values—i.e., fuzzy set theory [Motro 1990], the VAGUE system [Motro 1989; 1990], heuristic approach [Wang and Madnick 1989], and probabilistic frameworks [Fuhr 1990]. Handling imprecise data values gives users flexibility in defining their queries. If the user is unsure of the precise values desired or is not sure if the precise values are available, the system can return all relevant information. In several of the aforementioned solutions, the system will rank possible responses before displaying them to the user. The SSM extends this capability to schema terms (the metadata that describes the data values maintained in the database). Just as data values can be imprecise or vague, the terms used to access the data can be imprecise or vague. The SSM can return all relevant sources of data in response to a user's query.

3.4.2 *Semantic Distance.* A key aspect of handling imprecise data is defining the degree of similarity between two values. Numerical similarity is easy to determine—just take the difference of the two numbers. Semantic similarity between concepts (terms) is more complex. Roget's goal in creating his thesaurus was to define a linear ordering of English words so that semantic similarity would correspond to numeric similarity (just count the number of words between two terms [Dutch 1965]. He did not accomplish this goal because word meanings are too complex to arrange linearly.

Some work on defining semantic locality has been done in the context of the National Library of Medicine's Unified Medical Language System (UMLS) [Nelson et al. 1991]. Semantic locality has at lest four different aspects in the UMLS: synonyms and lexical variants, semantic types, contextual information, and co-occurrence of terms. UMLS entries include synonyms and variant spellings/usages of terms. These are assumed to be semantically identical. UMLS terms can have one or more associated semantic types (broad categories of similar concepts). If the set of types for two terms overlap significantly, the terms are semantically similar. UMLS terms can be defined based on their position in a hierarchical classification scheme such as the National Library of Medicine's Medical Subject Headings hierarchy. Terms in the same subtree are semantically similar. Finally, co-occurrence of terms measures

the number of times two terms are found together in sample journal articles. This is similar to the information retrieval system analysis described above.

4. SUMMARY SCHEMAS MODEL

The Summary Schemas Model (SSM) is proposed as an adjunct to multidatabase language systems that provides automated support for identification of semantically similar entities. This automatic identification allows the SSM to accept imprecise user queries. An imprecise query describes the data to be accessed using user-defined terms rather than system-defined terms. SSM imprecise-query processing matches the user-defined terms to semantically similar system terms. The imprecise-query processing is made possible by creating an abstract view of the available data from local databases in a hierarchical structure—i.e., a hierarchy of summary schemes. A summary schema is an abstract, yet concise description of the data available in a group of input schemas. It is not an exact representation of the data from input schemas, but it does retain most of the semantic content of those schemas. Summary schemas are created by using the hypernym links from an online-system taxonomy.

Multidatabase language systems are the target environment for the SSM because they represent a more suitable design approach (than global-schema multidatabases) for large systems or for distributed systems which allow small machines to participate. Although the SSM does entail a global data structure (the summary schemas hierarchy), this data structure is smaller than a global schema, and its creation and maintenance are partially automatic (unlike global schemas). However, the concepts and benefits of the SSM could be extended to other types of global-sharing systems (e.g., as an aid to global-schema design in global-schema multidatabases) which have the problem of trying to identify semantically similar entities despite differences in representation and naming.

The SSM is proposed based on the relational data model. Relational database technology is well defined, mathematically rigorous, and popular in the marketplace. Most current multidatabase projects use the relational model and SQL-like access languages for their global structure. However, today some multidatabase researcher are exploring data models with more advanced semantics, particularly, object-oriented models [Fankhouser et al. 1988]. Since the input to the SSM is just a group of access terms describing available data, SSM concepts could easily be extended to object-oriented multidatabases when they become more widely used. In this case, an interesting problem arises because object-oriented systems place some of their semantics in the methods attached to an object. Methods represent functions rather than data—verb terms instead of noun terms. The SSM relies heavily on the semantic relationship of hypernymy between terms. Verb hypernymy has different linguistic characteristics than noun hypernymy [Collier and Fellbaum 1988; Ravin 1990]. It is unclear (at this time) how the semantic identification and interpretation properties of the SSM would be extended to include verbs.

4.1 System Taxonomy

The semantic power of the SSM comes from the linguistic knowledge represented in an online-system taxonomy. This taxonomy combines information traditionally found in dictionaries and thesauruses. The taxonomy has an entry for each disambiguated definition of each term from a general lexicon of the English language. Each entry has a precise definition of the term and semantic links to related terms. The semantic links used in the SSM taxonomy are synonymy and hypernymy/hyponymy. Synonym links are symmetrical. Hypernym and hyponym links are reciprocal. The taxonomy is hierarchical in structure with multiple top-level nodes and some cross links between hierarchies at lower levels. Hypernym/hyponym relations are the hierarchy links of the taxonomy, while synonym relationships are the cross links between hierarchies or between leaf nodes at the lowest level.

4.1.1 *Taxonomy Characteristics.* Key aspects of the SSM taxonomy are a general lexicon, disambiguated entries, a simple hypernym hierarchy, semantically intuitive hypernyms, and limited synonym cross references. A general lexicon is necessary to represent the potential range of information and to support the potential range of user query terms. Disambiguated entries allow precision in specifying relationships between terms and in defining specific entities. The simple hypernym hierarchy and limited synonym cross references make the taxonomy structure easier to traverse and to calculate paths of semantic links between terms. This is important for calculating Semantic Distance Metric values (Section 4.3). An intuitive hypernym is a more general term which obviously captures the semantics of its hyponyms, as opposed to an obscure term which is technically correct, but is not well known.

One goal of our research has been to find an existing taxonomy that meets the SSM requirements rather than constructing a new taxonomy. Constructing a general taxonomy is beyond the scope of this work. Moreover, it would be a significant research effort in itself and would require significant linguistic knowledge and skill. By using an existing taxonomy, or combination of existing dictionaries and thesauruses, we have confidence that the definitions and semantic relationships represented have been tested and found to be semantically useful. Therefore, our linguistic base is solid.

Two existing taxonomies were explored for use with the SSM [Bright and Hurson 1991]. The first was the 1965 version of *Roget's Thesaurus*, and the second was a taxonomy derived from *Webster's 7th New Collegiate Dictionary* [Gove 1963; Dutch 1965]. These taxonomies were used to derive summary schema hierarchies from sample database schemas representing approximately 1000 access terms. Using these two taxonomies for schema summarization allowed us to judge their suitability for SSM processing. The Roget taxonomy only exists in text form (later versions are online but do not contain the hypernym links), so the relevant portions had to be manually entered for automated processing. The Webster dictionary is online, and the taxonomy was created using automated parsing tools [Byrd 1989; Klavans et al. 1991a; 1991b; Ravin 1989]. Despite its flaws, Roget's taxonomy was judged to be more suitable to the needs of the SSM at this time [Bright and Hurson 1991].

Therefore, all SSM processing is described in terms of Roget's structure. However, the work by IBM's Lexical Systems Group holds much promise, based on ongoing research and stated future directions.

4.1.2 *Taxonomy Implementation.* The full system taxonomy consists of terms, their disambiguated definitions, and the semantic relationships between the definitions. The format of a full taxonomy entry is: "term," "term definition text," pointer to hypernym, list of hyponym pointers (not included for leaf terms), list of synonym pointers (not included for nonleaf terms in Roget's taxonomy). Note that Roget does not include term definitions; therefore, it must be supplemented with a corresponding dictionary to be fully enabled for the SSM. The full taxonomy must be available online so database designers and users can understand the precise meaning of system access terms. Access terms are the names in database schemas that describe available data (e.g., database names, relation names, and attribute names in the relational data model). However, the SSM does not require the full taxonomy for its automated processing. Schema summarization and imprecise-query processing do not use term definitions or the lowest-level (entry-level) terms of the hypernym hierarchy. Definitions are only necessary for helping users understand the semantics of a term. Each entry-level term maps to a particular synonym list represented by a subhead. The synonyms in a subhead list are assumed to be semantically equivalent; therefore, automatic processing can consider the list a single unit (the subhead). Most of the storage required for the full taxonomy is consumed by the text definitions and the entry-level terms (the bulk of the terms are in the lower levels of the hierarchy). Also, the operational taxonomy is only used by the system, so terms can be identified by a numeric ID, which is smaller than the textual representation of a term. Hence, the operational taxonomy (for automated processing) is significantly smaller than the full taxonomy (Section 4.5.4). The format of an operational taxonomy entry is: term ID, pointer to hypernym, list of hyponym pointers (not included for leaf terms), list of synonym pointers (not included for nonleaf terms in Roget's taxonomy). Note that leaf terms in the operational taxonomy are the subhead terms.

The full SSM taxonomy is not a specialized structure. The semantic knowledge represented in the taxonomy is useful to a variety of other applications, including natural language processing [Byrd 1989; Klavans et al. 1991b]. Hence, the taxonomy can be shared, and its cost amortized over multiple users. The operational taxonomy used for automated SSM processing can be easily extracted from the full taxonomy.

4.2 Hierarchy of Summary Schemas

A summary schema is a collection of access terms which represents the semantic content of a group of input schemas. It represents the input data in a more abstract manner and consequently needs fewer terms to describe the information than the sum of the terms in the input schemas. The terms in a summary schema are the hypernyms of the terms in the input schemas. The semantic abstraction and reduction in numbers of terms results from the linguistic properties of the hypernym relationship. For example, consider two

base relations—one of which includes the attributes "city" and "zip code" while the other has "city" and "country." A global-schema representation of these schemas might have a generalized object with the attribute "city," but also retains specific objects with "zip code" and "country" attributes. The global schema is a precise representation of the base schemas. The summary schema for the same base relations may represent the input attributes with a single access term (hypernym) "location." "Location" retains the essential semantics of the "city," "zip code," and "country" as they are used in the base relations, but represents those semantics with a more general (abstract) term. The hypernym relationship allows a single summary schema term to represent three base access terms. A user interested in finding "location" information in a general sense may be satisfied with results from either base relation (assuming other requirements of the query can be satisfied by the relation). Therefore, the user can compose the query using any term (meaningful to the user) which maps to "location" in the system taxonomy. A user who needs a specific representation of "location" (e.g., "city," "zip code," rather than "city," "country") can find it using his/her own term, by using the Query Refinement Facility (Section 4.4.2).

The SSM structures multidatabase nodes in a hierarchy. Each internal node in the hierarchy contains a summary schema created from the schemas of its children. For simplicity, we consider only leaf nodes to contribute actual database schemas. Internal nodes contain only summary schemas. The summary schema at each internal node represents an abstract view of the information available in its subtree. The hierarchy is kept fairly short (five levels in the Roget taxonomy) in order to minimize message passing during imprecie-query processing. For a multidatabase system with many nodes, this means the hierarchy will be very bushy. Each internal node also contains a copy of the operational taxonomy. The internal nodes are responsible for most SSM processing.

4.2.1 *Schema Summarization Process.* Creating the summary schemas hierarchy requires some initial input from local database administrators, but the rest of the creation and maintenance process is automatic. At each leaf node, the local DBA must map the terms in the local schema (which is to be shared with the multidatabase) to entry-level terms in the system taxonomy. Since the SSM taxonomy contains a general lexicon, all local-access terms should correspond to an entry-level term. The effort required by each local DBA is minimal since she/he is assumed to know what "local-access terms" mean. Terms can be automatically looked up in the full system taxonomy, and the only decision by the DBA will be which disambiguated definition to use.

The entry-level terms in the taxonomy are entries in the synonym lists for subheads. At the first internal level of the hierarchy, a summary schema consists of subhead terms corresponding to the entry-level terms from the child leaf-nodes. The hypernym relationship from the system taxonomy automatically specifies the subhead for each entry-level term. At each successively higher level in the hierarchy, the summary schema consists of the list

of hypernyms corresponding to the terms in the previous lower-level schemas. Figure 1 shows a partial summary schemas hierarchy. Each internal node takes the list of access terms from its children's schemas, uses the system taxonomy to map to the hypernyms, and keeps those hypernyms as its summary schema. Each term in the summary schema has a list of pointers to child nodes which contain hyponyms of the term. These downward pointers are used to search for lower-level (more concrete) terms.

When the summary schema's hierarchy has been created, maintenance is also automatic once changes at the leaf nodes have been mapped to entry-level taxonomy terms. Adding a new leaf will cause terms to be added to ancestor summary schemas until a summary schema is encountered that already has the hypernym ancestor. Deleting a leaf term will cause ancestor summary schemas to delete hypernyms up to the point where the hypernym represents a term from another lower-level schema, as well as the deleted term (i.e., the list of child node pointers is larger than one). Hence, most changes will be restricted to a particular subtree of the overall hierarchy. Maintenance is required when nodes connect to the multidatabase, leave the multidatabase, or change the schema being shared with the global system.

4.2.2 *Implementation of the Hierarchy.* The system hierarchy for the SSM is a logical partition of nodes, but it will usually correspond to fast underlying physical network connections. Parent-children links should have underlying physical pathways with high-performance and low-propagation delays for message passing. Large networks frequently have a physical hierarchy of nodes. Leaf nodes are typically linked by a local-area network (LAN). Each LAN has a designated node that acts as a gateway to a higher-level network which consists of gateway nodes from other LANs. This higher-level network may in turn have a gateway to an even higher-level network, and so on. The higher-level networks are sometimes called backbone networks and usually have high-performance characteristics in order to handle the voluminous traffic flow between lower-level networks. The logical hierarchy of the SSM would typically be mapped directly onto the corresponding nodes in an existing physical hierarchy, i.e., the gateway for each network level would be a natural site for the parent node function of the SSM hierarchy. For example, nodes A, B, and 4.A in Figure 1 could be machines on the same LAN. Assuming Node 4.A was the LAN gateway to a higher-level network, Node 4.A would be the best choice to maintain the summary schema for databases on the LAN. However, the SSM logical hierarchy could be mapped differently as long as the parent-children links had good performance.

The SSM has a concept of closeness between two nodes in the system. The measure of closeness is the measure of the performance of the network path between the two nodes, i.e., close nodes have high-performance/low-propagation delay paths, while distant nodes have low-performance/high-propagation delay paths. The closeness measure does not need a precise quantification since it will be used mostly for comparison purposes rather than strict measurement. Node-processing capacity is also a consideration in assigning positions in the hierarchy. Nodes at higher levels in the hierarchy have more

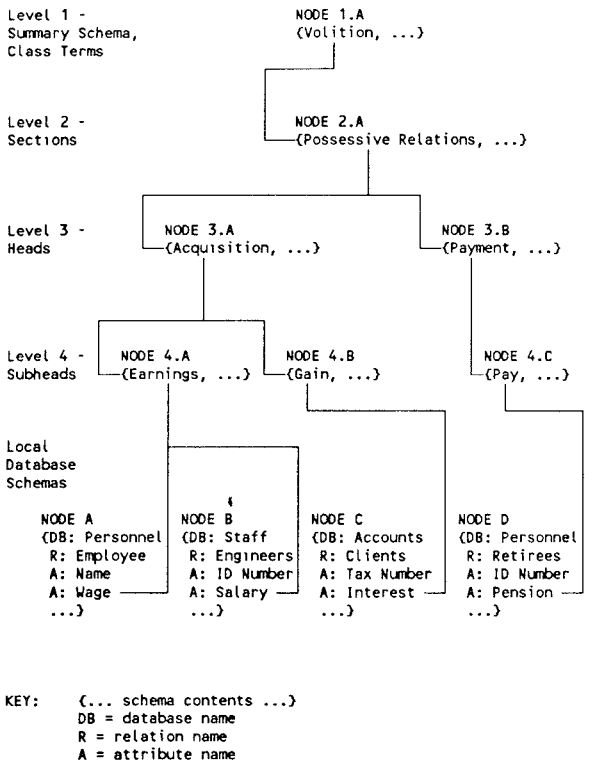


Fig. 1. Sample schema hierarchy with summarization of selected terms.

data and processing overhead to manage, so larger capacity nodes should be selected. Specialized database machines may be appropriate at internal nodes in order to provide efficient global query processing [Hurson et al. 1989; 1990]. Leaf nodes only have to maintain their own export schemas and have minimal global-processing responsibilities.

There are a number of advantages of partitioning SSM nodes in a hierarchical fashion. First, such an organization is a common approach for subdividing a large problem into manageable pieces. Each node in the SSM hierarchy has information about all the data available in its subtree. Since the global data representation is distributed, the task of managing that representation is also distributed. Upper levels of the hierarchy will be responsible for more information (larger subtrees), but these will generally be larger nodes with more machine and human resources available to shoulder the load. The SSM hierarchy can be mapped to represent the organization of the entity(ies) that own(s) the multidatabase. By matching organizational lines, data flow can be localized and will typically stay within a subtree. It is likely that users will most frequently use data close to themselves in the organization, hence, close to their node in the SSM hierarchy. Performance will be better when queries can be satisfied by close network nodes.

Finally, an unequal distribution of function and global responsibility allows small machines or machines with little available capacity to participate in the multidabase as leaf nodes. Many existing multidatabase systems require equal distribution of global data and processing [Hurson and Bright 1991a; 1991b]. However, today's distributed systems typically contain machines that vary widely in their capacity and performance. The system will provide better performance if global function is distributed according to each node's capacity and capabilities.

4.3 Semantic-Distance Metric

A key feature of the SSM is the ability to identify semantically similar entities. The Semantic-Distance Metric (SDM) provides a quantitative measurement of "semantic similarity." If there was a linear order of words by meaning [Dutch 1965], the semantic distance between meanings would be a simple measure—just count the number of intervening terms. Similarly, the value difference between two numbers is quantified by the number of positions between them on a number line. However, the linguistic complexities of word meanings do not fit a linear order. Roget's eventual taxonomy was a hierarchy of hypernyms with synonym cross references between subtrees. Just as Roget's taxonomy is more complex than a linear order, the SDM is a measure of semantic similarity which is more complex than a simple count.

All pairs of terms in the SSM taxonomy are connected by some combination of hypernym/hyponym and synoym links. The SDM is a weighted count of the links in the path between two terms. Terms with only a few links separating them (a small SDM value) are semantically similar. Terms with many links between them (a large SDM value) have less similar meanings. Link counts are weighted because different relationships have different implications for semantic similarity. In the SSM taxonomy, for example, synonym links would have a lower weight than hypernym/hyponym links because synonymy is a more precise indicator of semantic similarity. The SDM is defined in Figure 2. For Roget's taxonomy as used in the SSM, the summation in the SDM calculation will have two terms—one for synonym links and one for hypernym/hyponym links. The SDM calculation is extensible to more sophisticated taxonomies (with more semantic relationships) by adding more terms to the summation. An example of applying the SDM to find semantic matches for "income" is shown in Figure 3. In the example, SDM values of 1 to 3 yield semantic matches which are fairly specific to "income" in the sense of compensation for specific work. However, an SDM value of 4 yields terms that are still "income" in the sense of money received, but are not as semantically close to "income" in the sense of work compensation.

The SDM is intended to be a simple calculation that represents linguistically significant information. Finding the shortest path between nodes (terms) in a graph (taxonomy) is a well-understood problem. However, the SDM calculation is performed frequently during imprecise-query processing (Section 4.4), so the emphasis is on defining a fast calculation. An SDM value is intended to quantify the semantic similarity between two terms. The

LC - number of links between 2 terms

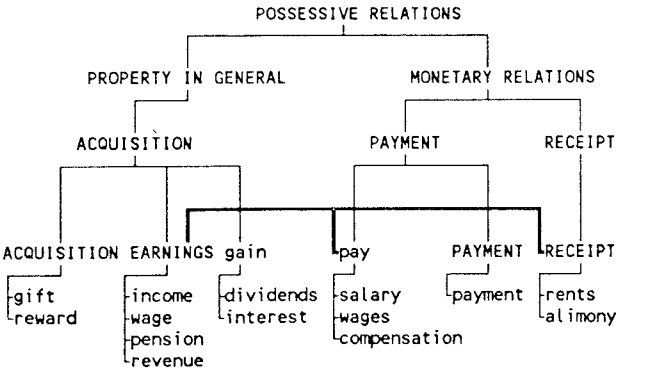
LW - weight (relative importance) of a link

i - represents a particular type of link, LW and LC will be different for each type of link

SD - semantic distance between 2 terms, the lower the value the closer the terms are in meaning

$$SD = \Sigma (LC_i * LW_i)$$

Fig. 2. Semantic-distance metric formula.



KEY: – hypernym/hyponym links
– synonym cross links
lower case terms are terms found in some local schema
UPPER CASE TERMS ARE HIERARCHY TERMS
all terms in bottom level lists are implicit synonyms

GIVEN: LW = 1 for synonym and hypernym/hyponym relations

Terms with SD = 1 for imprecise reference "income":

wage, pension, revenue

Terms with SD = 2 for imprecise reference "income":

pay

Terms with SD = 3 for imprecise reference "income":

gain, salary, wages, rents, alimony

Terms with SD = 4 for imprecise reference "income":

dividends, interest, payment, gift, reward

Fig. 3 Sample semantic-distance metric calculations.

linguistic correctness of the SDM is based on the linguistic correctness of the relationships used in the calculation and the relative values of the weighting coefficients (synonymy more important than hypernymy/hyponymy). Semantic similarity is a subjective judgment, and the semantic relationships assigned in a taxonomy (such as Roget's) are subject to the knowledge, experi-

ence, and bias of the taxonomy compiler. Therefore, the semantic matches returned for a specific reference and a specific SDM value will have an element of subjectivity. For queries which require more objective control over the response, the Query Refinement Facility (Section 4.4.2) allows users to interactively find precise data references.

The simplicity of Roget's taxonomy limits the sophistication of current SDM calculations. Additional semantic relationships would increase the information content of the taxonomy and increase the complexity of the structure. More relationships would allow more terms in the SDM calculation. More sophisticated relationships may also allow different SDM formulas.

4.4 Imprecise-Query Processing

Standard multidatabase language system queries have precise specifications of data to be accessed and how to manipulate that data. A precise data reference includes location and the local-access term. The query origin node parses the query, sends data access requests to remote data sources, and combines the data accessed according to the operations specified in the query. An imprecise data reference does not have a location, and the access term does not necessarily represent an exact system access term.

Imprecise-query processing in the SSM performs the same basic steps as precise-query processing, but adds a reference resolution phase between parsing the query and sending the remote-access requests. If the user is unsure of the existence, location, or local-access term for a piece of data, she/he can describe the data in her/his own words and mark the reference as imprecise. The user is still responsible for defining the operations to perform on the accessed data. When an imprecise reference is detected at the query origin node, the summary schemas structure is used to match the reference to a semantically close precise reference (location and local-access term), and query processing proceeds normally with precise data references. The user can control the semantic matching by specifying an SDM value for the imprecise reference. A small SDM value requires the system to provide a close semantic match, a larger value gives the system wider latitude in finding a corresponding precise data reference.

Imprecise-reference resolution is essentially a search of the name space represented by the summary schemas hierarchy. The goal of the search is to find a matching term within the specified semantic distance of the user's term. For simplicity, we only consider queries submitted at leaf nodes. The search starts at the query origin node and searches up the system hierarchy for a node with a possible matching term in its summary schema. At this point the match is only "possible"—it is not positive because interior hierarchy nodes have hypernyms rather than actual access terms in their summary schemas. From a possible match node, the search goes down the subtree rooted there to look for an actual access term within the specified semantic distance.

A general algorithm for imprecise-query processing is given in Figure 4. This general strategy can be used to process imprecise references in a query or to return all possible matches for an imprecise reference. Specific

```
Step 1) At query origin node - parse query;
                IF all data references are precise THEN go to Step 4)
                ELSE send query to next higher node in system hierarchy;

Step 2) At next higher node -
                FOR each imprecise data reference DO
                        calculate SDM using imprecise data reference
                                and local summary schema as inputs;
                        IF any local term is within semantic distance THEN
                                send message down the hierarchy to see if
                                (               corresponding access terms from leaf node
                                                schema are within the semantic distance;
                                        IF so THEN replace imprecise with precise
                                                reference
                                        ELSE reference is still imprecise;

Step 3) IF all data references are precise THEN go to Step 4)
                ELSE IF this is the top level in the hierarchy THEN
                        reject the query because no data in the system
                        within the specified semantic distance
                ELSE send query to next higher level node and continue with
                        Step 2);

Step 4) all data references are precise, so execute query with standard multidatabase language facilities;
```

Fig. 4. General imprecise-query-processing algorithm.

implementations of these alternatives are discussed below along with possible optimizations of SSM query processing. Figures 5–7 show an example of SSM structure and processing. Figure 5 shows two sample databases and the relevant portions of the system taxonomy. Figure 6 shows the summary schema hierarchy based on the sample databases. Figure 7 steps through imprecise-query processing for a query based on the sample databases.

4.4.1 *User Queries.* A primary goal of the SSM is to automatically interpret imprecise references in a user's query and allow the system to process the query. When an imprecise reference is detected, the system checks the local schema for a corresponding term. If the reference can be satisfied locally, the imprecise reference is replaced with the local-access term and the imprecise-reference resolution phase of query processing is complete. If the reference cannot be satisfied locally, the imprecise term is mapped to an entry-level term in the system taxonomy. The hypernym for the term is extracted from the system taxonomy, and a message is sent to the leaf node's parent. The message contains the query origin node ID, the original user term, the term's hypernym, and the specified SDM value.

Upon receiving a search message from a child, a parent node checks its summary schema for a matching hypernym term. If there is a matching hypernym, then a downward-search message is sent to the first child with a corresponding hyponym. A downward search message contains the origin node ID, the original user term, the hypernym term in the upward message, and the specified SDM value. If that downward search message returns a precise data reference, the precise reference is returned to the origin node, and reference resolution is done. If the downward search message does not find a matching precise reference, the parent sends a downward search message to the node with the next corresponding hyponym (multiple nodes may have hyponyms mapping to the same hypernym at the current level). If no downward search messages produce a match, then the hypernym in the

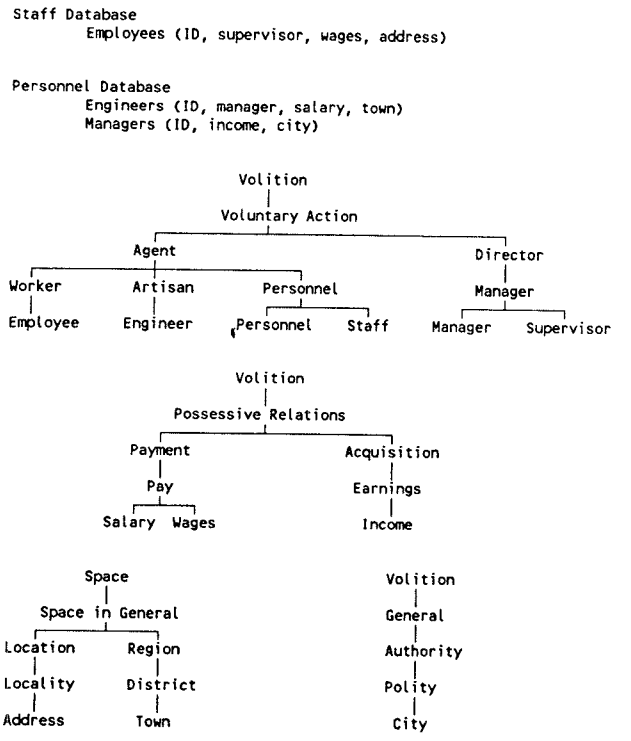


Fig. 5. Sample databases and partial taxonomy of access terms.

```
NODE 1 (leaf):  external schema -
        (staff, employees, supervisor, wages, address)

NODE 2 (leaf):  external schema -
        (personnel, engineers, manager, salary, town, income, city)

NODE 3:  First level internal node summary schema -
        (personnel<1,2>, worker<1>, manager<1,2>, pay<1,2>, locality<1>,
        artisan<2>, district<2>, earnings<2>, polity<2>)

NODE 4:  Second level internal node summary schema -
        (agent<3>, director<3>, payment<3>, location<3>, region<3>,
        acquisition<3>, authority<3>)

NODE 5:  Third level internal node summary schema -
        (voluntary action<4>, possessive relations<4>, space in
        general<4>,      general<4>)

Note - the numbers in <,> are pointers to lower level nodes with corresponding hyponyms.
```

Fig. 6. Example of summarization results.

original message is mapped to the next higher-level hypernym and the message is forwarded to the next parent node. Each successive parent level performs similar processing until a match is found or the top level of the hierarchy is encountered. If no match is found at the top level, then there is no matching precise reference in the multidatabase.

```
GIVEN:
The SDM is a simple count of links in the hypernym hierarchy.  The SDM value used is 1.

The user is somewhat familiar with the STAFF and PERSONNEL databases and wants to retrieve employees that
make more than $20,000.  The user is unsure of the access term for wages in Node 2.

The LET command in the query represents the multidatabase language system ability to combine multiple data
references into a single term.


QUERY:
LET PERS = NODE1.STAFF AND NODE2.PERSONNEL;
        /* open two databases with precise data references */
LET EMP = NODE1.EMPLOYEE AND NODE2.ENGINEERS;
        /* combine two precise relation references */
LET PY = NODE1.WAGES AND pay;
        /* "pay" is an imprecise data reference */
SELECT  ID,PAY
FROM  PERS.EMP
WHERE  PY > 20000;


QUERY PROCESSING:

Step 1)  At Node 1 parse the query.  PY(pay) is an imprecise reference so pass the query to Node 3.

Step 2)  Node 3 has a summary schema term "pay" which is 0 links away from PY(pay).  Since the SDM at Node 3
is less than 1, send a message down the hierarchy to see if the actual access term at Node 2 is within the
specified semantic distance.

At Node 2, the access term "salary" is 1 link away from "pay" (the hypernym link).  Replace PY(pay) with
PY(NODE2.SALARY) and return to Node 3.

Step 3)  All data references are now precise.

Step 4)  Execute the multidatabase query.
```

Fig. 7. Example of SSM query processing.

Upward search message processing involves exact hypernym matching, while downward search message processing uses the SDM. When an internal hierarchy node receives a downward search message, it finds which local summary schema terms map to the hypernym specified in the message (i.e., the hypernym that caused the message to be sent). The node checks to see if any descendants of these terms are within the specified semantic distance of the user's imprecise term. For each local term that may contain matching descendants, the downward search message is propagated serially to the appropriate children (with the hypernym replaced by the corresponding local hyponym). As soon as one child returns a precise reference, the search is terminated, and then the precise reference is returned to the parent node. If no downward search is successful, the node reports failure.

When a downward search message reaches a leaf node, the SDM calculation can be applied directly to the user's imprecise term and available local terms. Only this level of the search is capable of making positive matches and returning precise references to replace the user's imprecise terms.

This implementation of the SSM imprecise-reference resolution process returns the first precise reference encountered within the specified semantic distance. However, there may be a reference in the multidatabase which is a more exact semantic match, but happens to be in a portion of the system hierarchy which has not been searched. There is a tradeoff between finding

an answer quickly (the closest one in the network) and finding a more exact answer.

If the user wants a more precise semantic match, she/he just has to specify a smaller SDM value. If the user wants to know just what the range of possible matches is, she/he can use the Query Refinement Facility to return all matches in the system.

4.4.2 *Query Refinement Facility.* System designers and application writers will often be interested in the full range of possible matches for an imprecise data reference. Finding all semantically similar entities in the global system is the first step in the schema integration process. Programmers may want to know all possible responses before coding specific queries in an application. The Query Refinement Facility (QRF) returns all available system matches for a given imprecise reference. QRF imprecise-reference resolution is similar to the process for user queries discussed in Section 4.4.1, but it does not stop after the first match is found. Also, QRF processing does not perform any actual data access; it simply returns the precise schema terms (i.e., it stops after the imprecise-reference resolution phase). Upward search messages are propagated to the top of the hierarchy. All possible downward search messages are generated. All precise matches found at leaf nodes are accumulated at the QRF origin node.

When all system references within the specified semantic distance of the user's reference have been accumulated, they are displayed to the user in a list which is ordered from closest semantic match to most distance. The list displays the matching, precise system term, its location, and its semantic distance value. If the user wants to refine the query further, she/he can change the imprecise reference or modify the SDM value to restrict or enlarge the search.

In a multidatabase system, any particular piece of information may be partitioned and/or replicated in multiple places in various forms. Some user queries can be satisfied with a single instance of the data item. For example, if a user wants to know an address, any copy of the address can be used to satisfy the user's query. However, in other cases, a user query will require all (or some subset) of the partitions/replications to be combined in order to properly respond. For example, if a user wants to know the total income of a person, then the person's salary, bank interest, investment returns, etc. must all be retrieved and combined in order to calculate the answer. Imprecise-query processing finds a single match for an imprecise data reference and uses that match to respond to the query. However, in some cases multiple matches might be required in order to respond correctly. The choice of whether a single instance or multiple instances of a data item is required to satisfy a user query depends on the semantics of the query and the intent of the user. It is not clear that these semantics and intent can be identified and/or understood by the system. Hence, although the imprecise-query–processing algorithm could be changed to retrieve multiple matches to an imprecise data reference, it is more suitable to expect users to use the

QRF when multiple matches are appropriate. The QRF allows users to identify multiple matches. Then the user can decide which matches are appropriate and resubmit the query with multiple, precise reference in place of the initial imprecise reference.

4.5 Benefits

4.5.1 *Semantic Identification.* As noted previously, identifying semantically similar entities despite representational differences in multiple input schemas is an important problem in schema integration. The SSM automates this process via the Query Refinement Facility. The semantic-distance value associated with each QRF response gives a quantitative measure of the semantic similarity. The dictionary entries and semantic links in the system taxonomy provide all the semantic information that can be automatically abstracted from remote schema terms. Hence, global-database designers do not have to have extensive knowledge about hundreds or thousands of local schemas. They are not required to spend inordinate amounts of time searching remote schemas.

Programmers have more control over the data being requested for application processing. The QRF displays the range of similar data available and allows an iterative search for the precise references required. Programmers can optimize for less precise data that is closer in the network, hence providing better performance, or can choose more precise references that may entail extra communication costs.

4.5.2 *Imprecise Queries.* The ability to submit imprecise queries is an important enhancement to existing multidatabase user interfaces. Users can retain their own world view and its corresponding vocabulary while accessing information from independent sources with differing vocabularies. They are not required to learn or search through large lists of system access terms. The system automatically interprets user requests based on the linguistic knowledge represented by a standard thesaurus and dictionary. Hence, semantic interpretations by the system are as accurate as the accepted, linguistic authorities allow.

An issue with automated system interpretation of a user's intent is the correctness of that interpretation. The interpretive ability of the SSM is limited by the precision of local-database designers in naming their data, by user's precision in specifying access terms, and by the limits of encoding semantic knowledge in thesauruses and dictionaries. Imprecise queries run the risk of incorrect resolutions or of resolutions which are not as semantically accurate as desired. This risk is inherent in allowing imprecise inputs. However, the problem is not limited to systems that allow imprecision. A precise query may not return the correct response if the user does not know the correct precise terms. Also, local-database designers may choose inaccurate precise access terms. Response correctness (for imprecise queries) can be improved by encouraging users and local-database designers to consult the online taxonomy when defining access terms. The taxonomy should help them

be more precise in defining their meaning. Also, the QRF can be used to iteratively refine access terms based on a ranking of system responses to the previous query. The terms and relationships in the taxonomy also affect response correctness. For example, using fully disambiguated definitions for terms (one term = exactly one meaning) based on current usage allows users to meaningfully relate terms and meanings. Providing more linguistic relationships among terms (different forms of hypernymy, for example) allows the system to be more precise in determining semantic distance. Formal models for semantic distance and response correctness are areas for future research.

4.5.3 *Global Data Structures.* The global data structures maintained by the SSM are smaller than a corresponding global schema. They are also created and maintained automatically. Roget's hypernym structure consists of 6 classes, 39 sections, 990 head terms, an estimated 6300 subheads, an estimated 215,000 entry-level terms (the lexicon), and an estimated 30,000 cross references between subhead synonym lists. The full system taxonomy contains all these terms, their associated taxonomic links, and their disambiguated dictionary definitions. Assuming an averge of 10 bytes per term name and 100 bytes per definition (a somewhat arbitrary estimate of a widely variable number), the size of the full taxonomy is on the order of 24.5 MB. However, the operational taxonomy (for schema summarization and imprecise-query processing) only contains the upper-level (subhead and above) term IDs and their taxonomic links. This totals to 7335 identifiers, 7335 two-way hypernym/hyponym pointers, and 30,000 cross reference pointers. Each ID and pointer is just an integer (4 bytes), so the total size of the operational taxonomy is approximately 208 KB.

The size of the summary schemas hierarchy is dependent on the number of local databases and the range of access terms used in their schemas. However, the number of hypernyms at each level of the system taxonomy represent an upper bound on the number of possible terms in each summary schema at the corresponding level. For example, a summary schema at the head level could not contain more than 990 terms. Each term in a summary schema has an identifier (corresponding to the term ID in the taxonomy) and a list of lower-level nodes that have hyponyms. The upper bound on the number of pointers in the node list is the number of children the current node has. Hence, a summary schema would normally be small enough to keep in main memory, too. One study measured the size of the summary schema hierarchy required to represent a set of sample local schemas compared to the size of the corresponding global schema [Sarma 1992]. The global-schema size (a union of the input schemas) was 41.5 KB. The total size of all summary schemas added together was 51 KB. Note however, that a global schema is replicated at each participating node, so the total size of the global schemas added together was 1.3 MB. Therefore, the total space (across the multidatabase system) required by the summary schemas hierarchy was only 4% of the total space required by the global-schema model.

5. EVALUATION OF THE SUMMARY SCHEMAS MODEL

The ability to accept imprecise user queries is a powerful feature for a multidatabase system. However, the cost of providing that feature must be quantified. A simulator has been developed to compare the overhead costs of imprecise-query processing to precise-query processing in a multidatabase language system [Sarma 1992]. A series of experiments was performed on sample database schemas to measure the performance of imprecise-query processing under a variety of conditions. Overhead costs for the QRF were also measured. The results showed that, on the average, imprecise-query processing adds little overhead cost relative to precise-query processing.

5.1 Summary Schemas Model Simulator

5.1.1 *Simulator Implementation.* The SSM simulator was written in Concurrent C and runs on a network of IBM PC RTs [Sarma 1992]. It models a multidatabase language system and the appropriate extensions for SSM imprecise-query processing (i.e., it supports the reference resolution phase). Each node in the multidatabase is represented by a separate process, and an arbitrary network structure is simulated through the use of sockets. Each node process reads a profile to determine its processing speed and its network connections. Leaf nodes in the SSM hierarchy (the only nodes with databases) read profiles that contain the local-database schema and the simulated-database size. The local-database size is specified by the number of tuples in each relation, the size of each attribute, and the average selectivity of attribute values. Measurements of local DBMS operations are calculated from the schema statistics—no actual data is maintained. The summary schemas hierarchy is generated from the local-database schemas. A single copy of (a subset of) Roget's taxonomy is maintained for use by all nodes. Although a hierarchy is imposed on the simulated network for imprecise-query processing, the underlying network links can have an arbitrary topology.

A major emphasis during simulator development was flexibility in defining the underlying multidatabase language system. Through profile values, the experimenter can control the simulated processor speed of each node, the network topology, the bandwidth of each network link, the size of each local database, and the local schema (access terms) of each database. The precise and imprecise queries used to run experiments are read from user-defined text files. For precise queries, the simulator emulates a standard multidatabase language system. Imprecise queries automatically trigger the SSM imprecise-query–processing extensions.

5.1.2 *Assumptions and Simplifications.* A number of assumptions and simplifications were made during simulator development in order to maintain a reasonable scope of work. The intent was to provide a robust experimental tool without sacrificing correctness or the general applicability of the results. The first simplification is that all simulated local databases are relational. Although a major characteristic of multidatabase systems is the

ability to integrate heterogeneous local data models, most current systems define some intermediate global data model and access language. The predominant choice is the relational data model. Restricting simulated databases to be relational merely eliminates one level of data model/access language translation from the standard multidatabase model. The simulator handles the primary relational operators—select, project, join, and union.

The simulator only accepts unary or binary query operations. A general multidatabase user interface must respond to complex queries with possibly nested suboperations. However, at some internal level of query processing, all queries are decomposed to unary or binary subqueries. Therefore, the simulator can produce results for complex queries, but the decomposition step must be performed manually. The schema summarization process and maintenance of the summary schemas are performed manually. Generation and maintenance of the summary schemas hierarchy does not impact the operational performance of imprecise-query processing; therefore, summary schema processing was deemed to be beyond the scope of current simulator development.

The SSM simulator performs little query optimization. Optimization techniques applicable to precise-query processing would also apply to imprecise-query processing after the query resolution phase. Hence, the performance gains would be similar in both cases. Since the purpose of the simulator is to measure the relative performance of imprecise- and precise-query processing rather than the absolute performance, query optimization was not a high priority. The importance of query optimization in an operational system is undisputed, but the associated development costs place it beyond the scope of the current work.

The SSM simulator does not implement the full power of the SDM. Imprecise-query processing only searches for semantic matches within an access term's subhead list. Searching for direct synonyms (terms within a subhead list) provides the closest semantic match to a user's request. The purpose of more sophisticated SDM calculations is to widen the semantic neighborhood to be considered in searching for a matching system access term. Thus, the current simulator provides the core semantic-identification function. More sophisticated SDM calculations are a subject for future extensions of the simulator.

5.1.3 *Simulator Operation.* The simulator is used to test the performance of representative queries that access various combinations of local and remote data. A set of queries is first submitted as precise queries—requesting relational operations on specific data at specific locations. The communications and processing costs of each query are recorded. The communications costs include the time required to transmit query-processing messages between nodes and the time to transmit the results back to the query origin node. Communications costs are based on the specified performance of the network links involved. Processing costs include the time required for query-processing operations and the time for performing the relational operations

at the local databases. Processing costs are based on the specified performance of the nodes involved and the size characteristics of the target databases. These measurements represent the baseline performance of each precise query in a typical multidatabase language system.

Next, each query is resubmitted with the precise data reference replaced by semantically equivalent imprecise references. The summary schemas hierarchy is searched to resolve the imprecise references before standard query processing is invoked. Imprecise-query processing involves sending search requests up and down the hierarchy and searching local summary schemas for semantic matches. The total communications and processing costs of each query are again recorded. The difference in the costs between the precise and imprecise version of each query represents the overhead of SSM processing. In each simulator experiment, the test queries are submitted at all SSM hierarchy leaf nodes, and the average cost over all query origin nodes is calculated. Averaging over all leaf nodes masks any local processing or structural aberrations and provides a more robust overall result.

5.1.4 *Sample Data.* Sample database schemas for simulator experiments were gleaned from a variety of database designs. The experiments used local databases with simulated relation sizes ranging from 1 KB to 500 KB. The calculated results sizes ranged from 0.01 KB to 700 KB. Network topologies, network link bandwidths, and processor speeds were chosen to be representative values drawn from standard texts. Network bandwidths ranged from 160 KB/second to 4.7 MB/second. For more information, interested readers are referred to [Bright and Hurson 1990].

5.2 Costs of Imprecise-Query Processing

A series of experimental cases were run on the SSM simulator in order to analyze the relative performance of precise- and imprecise-query processing. In each case, communications costs and processing costs are the measured basis for comparison. The first case repeats a few simple queries with various distributions of target schemas across the leaf nodes. The distributions vary from the best case for imprecise-query processing (semantic matches are available at every node) to the worst case (many semantically close schemas are available, but only one match). The second case demonstrates a variety of queries over a fixed distribution of local schemas. The queries vary in complexity, size of result, and how close the matching data is to the query origin node. The third case repeats the queries of Case 2, but measures the effect of a different network topology. The network is highly interconnected and provides more communications paths than just the SSM hierarchy links. The overhead costs of the QRF are analyzed in Case 4. Standard multidatabase systems do not have any function corresponding to the QRF, so the results are presented without reference to an existing baseline.

5.2.1 *Case 1—Best- and Worst-Case Performance.* In this experiment, a series of trials are run with a different distribution of local schemas for each trial. The intent is to quantify the performance of imprecise-query processing under best- and worst-case conditions. In each trial, a select, a project, and a

join query are submitted at each node. The same queries are used for each trial. All precise queries access data at node 0, while imprecise queries access data at the closest node with a semantic match. The network topology is shown in Figure 8.

In the first trial, a semantic match (for the imprecise query) is located at every leaf node in the network. This represents the best case for imprecise-query processing, since each can be resolved locally. For all nodes except node 0, the imprecise query is preferable to the precise query because the imprecise query provides a semantically correct result, but does not require any network communications overhead. In the next 15 trials, semantically matching schemas are removed, one node at a time, until only node 0 has a semantic match for the imprecise data references. Each of these subsequent trials (numbered 1–15) requires some imprecise queries to search farther in the network. In trial 15, both precise and imprecise queries are accessing the same database. However, the precise queries communicate with node 0 directly while the imprecise queries must search for the matching data.

In the next series of trials, only one local schema provides a semantic match for the imprecise data references, while a semantically similar schema is added to each leaf node in turn. This similar schema does not provide a semantic match for the imprecise references, but it does look like a match at higher levels in the summary schemas hierarchy. In other words, the similar schema maps to the same hypernyms as the target schema above the subhead level in Roget's taxonomy. This similarity causes imprecise-query processing to make extra searches (send extra messages) at internal hierarchy nodes, that end in failure at the leaf nodes. In the last trial, every leaf node has the similar schema, so queries from distant nodes (relative to the target schema) must do almost exhaustive searches through the summary schema hierarchy before finding the semantic match. Trial 30 represents the worst case for imprecise-query processing since there is only one semantic match in the system, but many false leads to weed through.

Figure 9 shows the communications and processing costs for the select queries over the 30 trials. Each data point represents the average cost (in microseconds) of all queries (one submitted at each leaf node). As expected, when imprecise-query processing can identify a local (or close) semantic match, average communications costs for imprecise queries are less than (or close to) precise queries that must access remote data (trials 0–15 in Figure 9a). In trials 16–30, the only semantic match for imprecise queries is at node 0. As semantically close schemas are added (trials 16–30 in Figure 9b), imprecise-query processing requires more search messages, and the costs rise. The cost of resolving the imprecision causes significant degradation in query-processing performance.

Similar series of simulations were conducted for the project and join queries (Figure 10). As expected, the average communications costs for precise queries are higher for project and join than for select queries. Project and join tend to return larger results; therefore, the communications costs for results dominate the costs for message passing. Thus, even in the worst case, imprecise-query–processing communications costs are only marginally larger

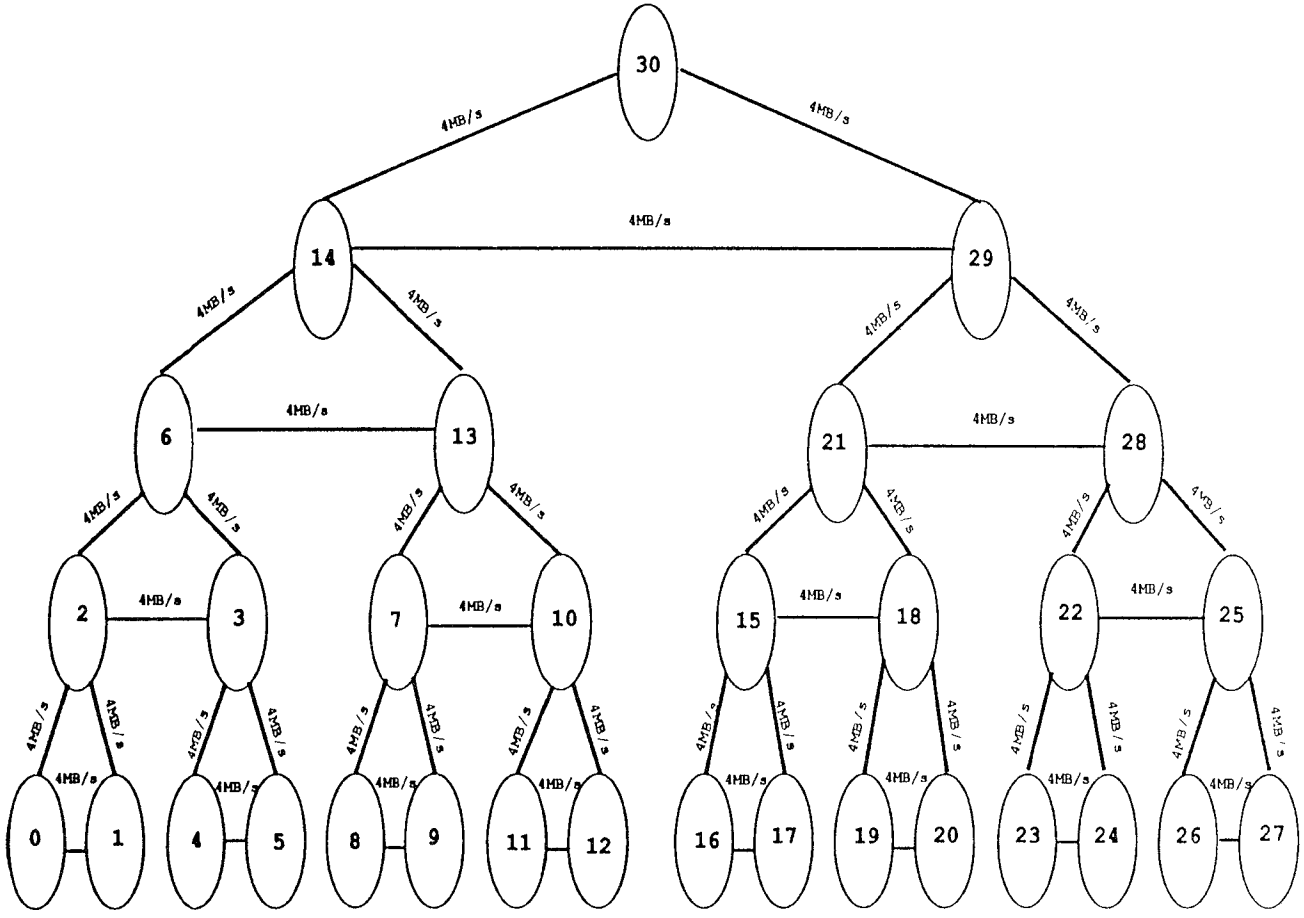


Fig. 8. Network topology for Case 1.

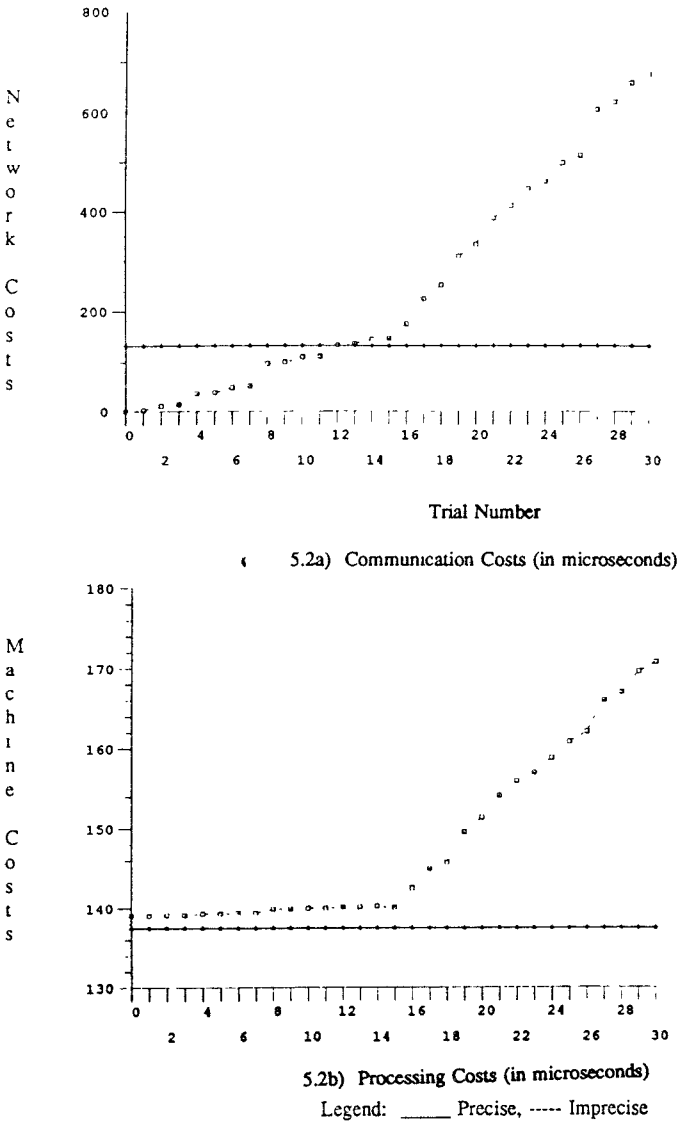
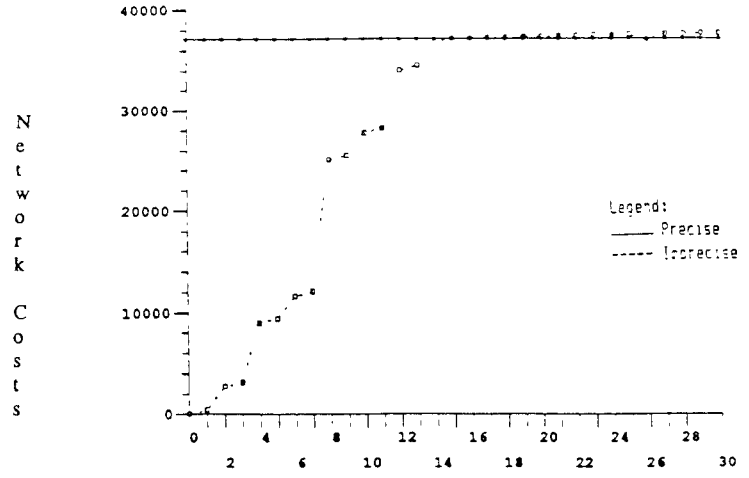


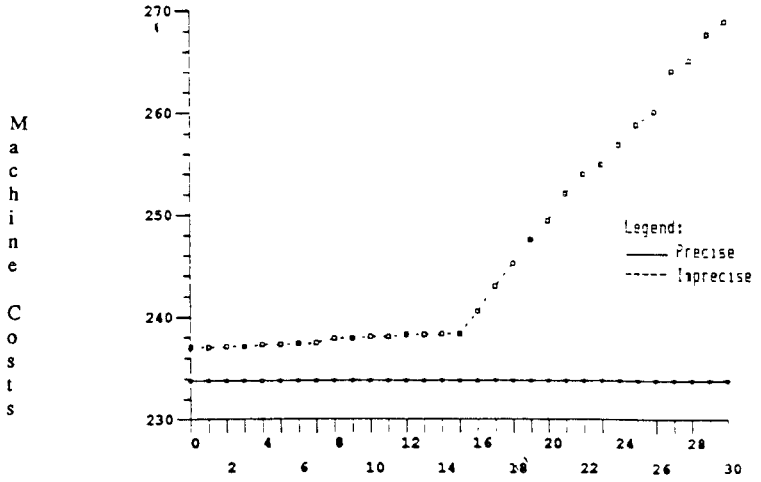
Fig. 9. Costs for selected queries—Case 1.

than precise-query processing. The processing costs curves are similar to select query results.

In summary, imprecise-query processing can save significantly on communications costs when the distribution of schemas is favorable. In the best case, imprecise queries required no communications costs and only marginally larger processing costs. However, with unfavorable schema distributions, imprecise-query processing adds communications and processing overhead. In the worst case, imprecise-query processing added over 500% in communica-



5.3a) Communication Costs (in microseconds)



5.3b) Processing Costs (in microseconds)

Fig. 10. Costs for join queries—Case 1.

tions costs for select (but very little for project and join). Worst-case processing costs showed that imprecise queries added between 16% (join) and 33% (project) overhead. In the average cases, precise and imprecise processing costs were relatively similar.

5.2.2 *Case 2—Average-Case Performance.* In this experiment, the schema distribution remains set while a variety of queries is submitted at each leaf

node. The queries represent different combinations of operations, target data, and result sizes. Each leaf node in the system submits the same set of queries and accesses data at all other leaf nodes in the system. The first set of queries (0–17) represents a set of select operations that return results ranging from 10 B to 1 KB. The second set of queries (18–24) is a mixture of projects and joins which returns results ranging from 5 KB to 700 KB. The intent is to show the robustness and scalability of imprecise-query processing over a wide range of queries (average-case performance). In general, it was expected that precise queries would cost less than imprecise queries. The flexibility of the simulator is shown by using a different network topology (Figure 11) than Case 1.

The average communications and processing costs for the first set of queries (0–17) are shown in Figure 12. Note that each query is independent, so the curves in subsequent figures are only useful for displaying the data points, not for depicting trends between data points. Despite the overhead of imprecise-query processing, imprecise queries cost less than precise queries in some cases (communications and/or processing costs). There are two reasons for imprecise queries to perform better than precise queries. First, as in some Case 1 trials, the imprecise query will locate a semantic match which is closer in the network than the corresponding precise data reference. Therefore, the imprecise query will incur less communications overhead. Second, the imprecise query may locate a semantic match in a database with different size characteristics than the precise data reference. If the imprecise query operates on a smaller database than the precise query, its processing costs will be less. If imprecise-query processing identifies multiple semantic matches for a query, the simulator automatically chooses the smallest database for processing. Naturally, such flexibility cannot be exploited by precise-query processing. The costs of the second set of queries are shown in Figure 13. There is no significant difference between imprecise- and precise-query processing. As in Case 1, when the query results are large, the result processing dominates the message-passing phase of query execution.

In summary, precise-query processing has only marginally better performance than imprecise-query processing over a wide range of queries and result sizes. Imprecise queries actually perform better than precise queries if there is a close semantic match or a semantic match at a smaller database. When the results are large, imprecise-query–processing overhead is negligible.

5.2.3 *Case 3—Impact of Network Topology.* The effects of network topology on performance measures were studied by adding more links to the network topology of Case 2 (Figure 11). The additional links form a chordal ring structure and were expected to aid precise-query performance by providing shorter path lengths for initial query propagation (imprecise-query processing still uses the hierarchy links for searching). The assumption is that an intelligent network manager always selects the shortest path for a message (the simulator does this). The scheme distribution and set of queries is

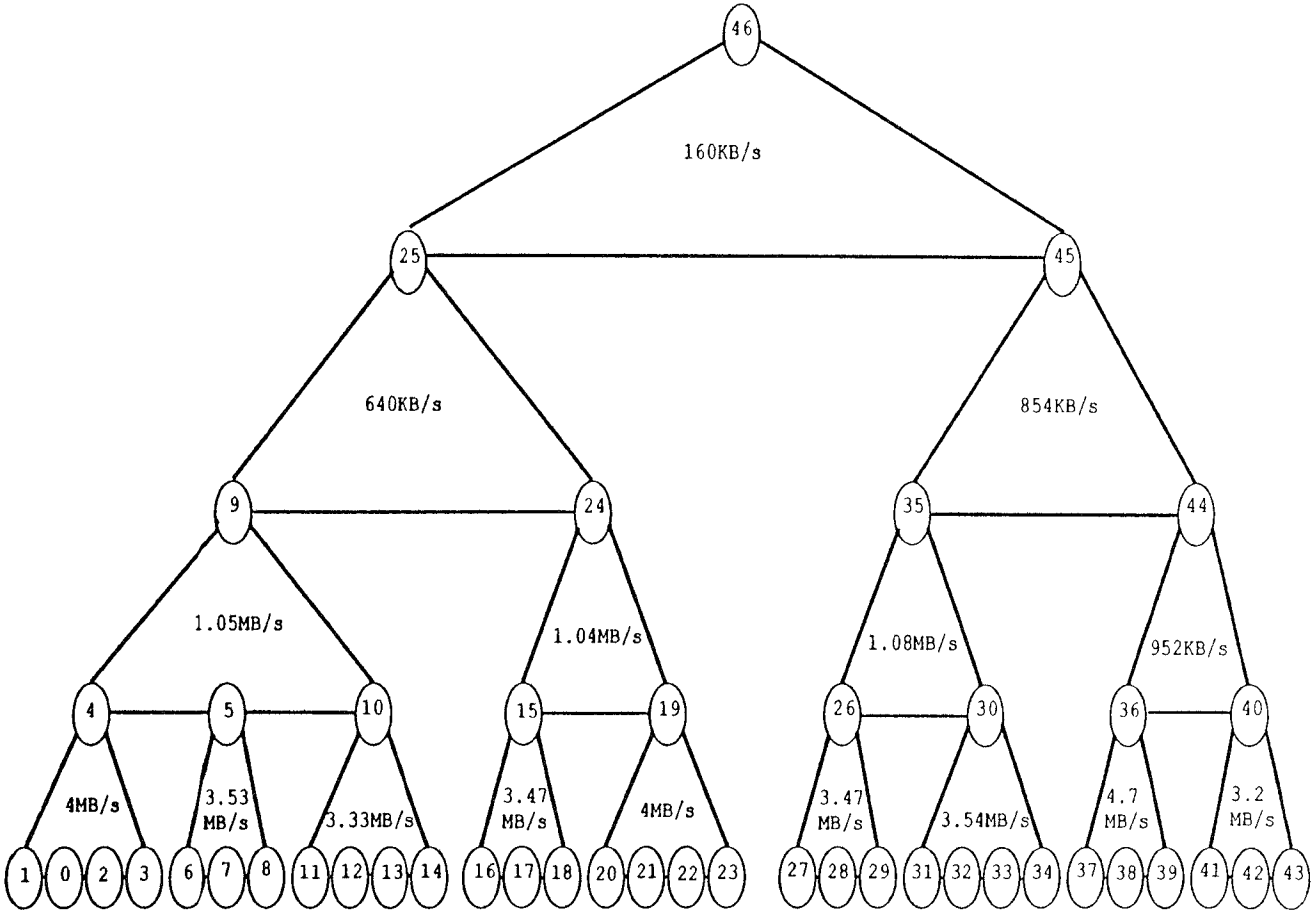
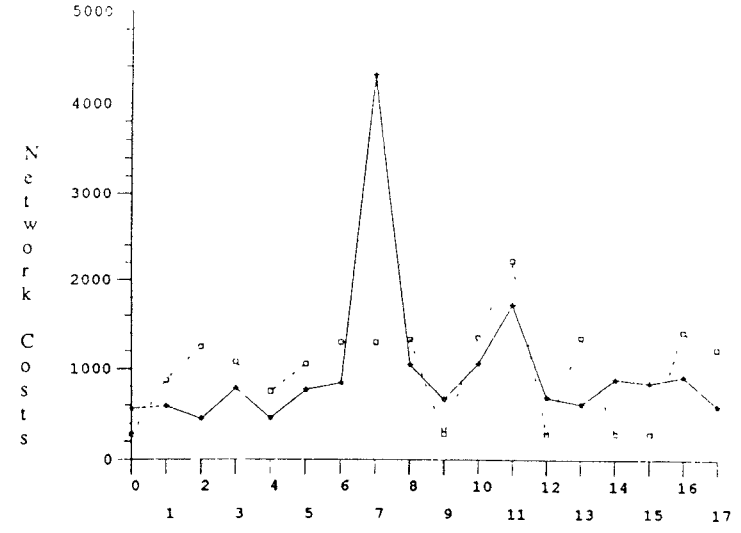
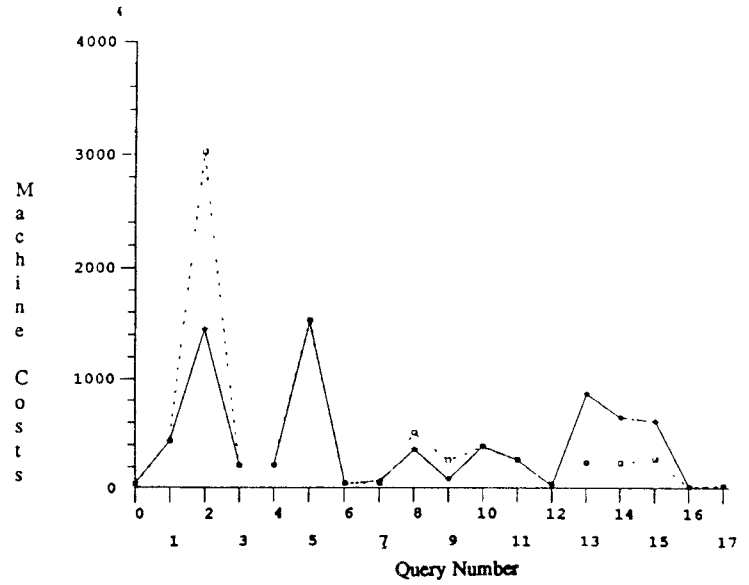


Fig. 11. Network topology for Case 2.



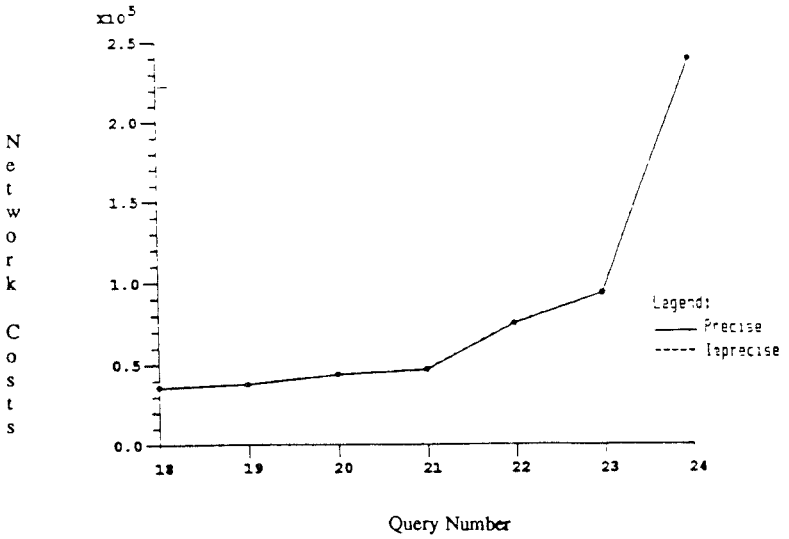
5.5a) Communication Costs (in microseconds)



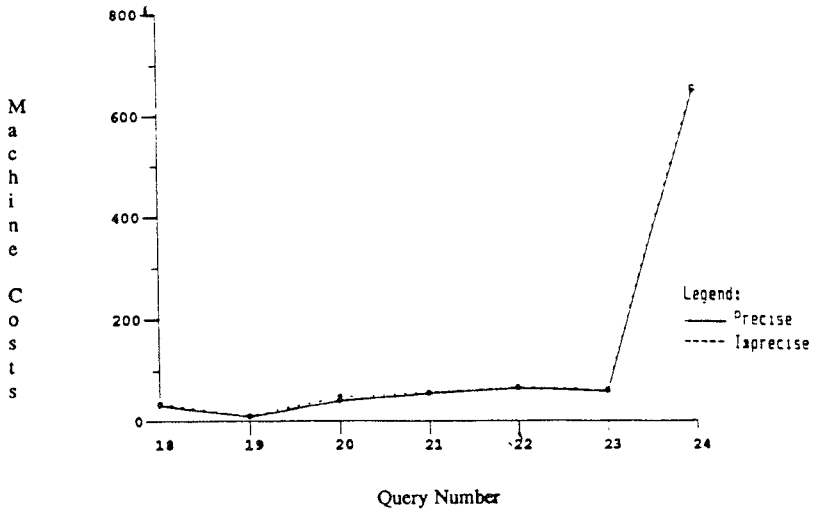
5.5b) Processing Costs (in microseconds)

Legend: _____ Precise, ----- Imprecise

Fig. 12. Costs for first set of queries (selects)—Case 2.



5.6a) Communication Costs (in microseconds)



5.6b) Processing Costs (in microseconds)

Fig. 13. Costs for queries 18–24 (projects/joins)—Case 2.

the same as Case 2, so any performance differences should be strictly due to network topology differences.

The processing costs for this case are identical to Case 2 (same enodes, same databases), so only the communications costs for the first set of queries (0–17) are shown in Figure 14. Again, the results are mixed, but the overhead

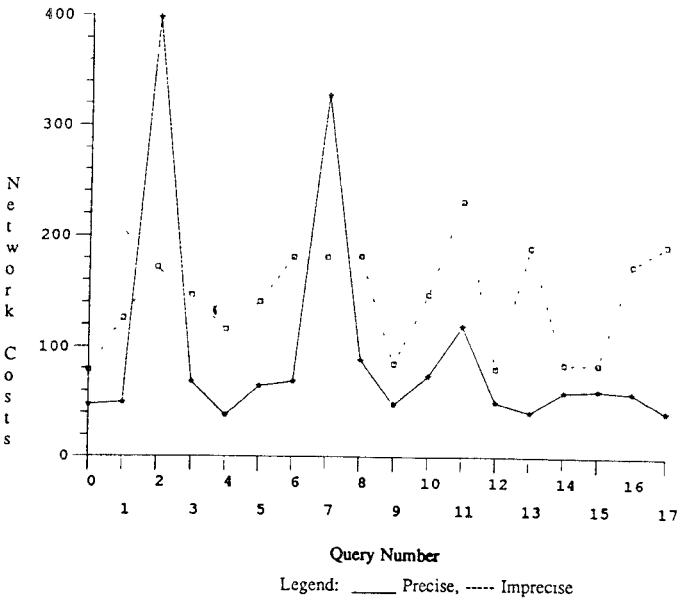


Fig. 14. Communication costs (in microseconds) for "select" set of queries—Case 3.

of imprecise-query processing is generally more pronounced than in Case 2. The results for the second set of queries (18–24) are not shown, since they again display almost no difference between imprecise and precise costs. In summary, a denser network topology does favor precise-query processing, but only to a minor degree. As noted in Case 2, when the result size is large, imprecise-query processing overhead is negligible.

5.2.4 *Case 4—Query Refinement Facility Overhead.* The final experiment analyzes the communications costs for the QRF. Since there is no corresponding capability in standard multidatabase systems, only the costs for QRF imprecise-query processing are reported. The network topology used is the same as Case 2 (Figure 11). There are no processing costs (only communications costs) since the QRF does not actually access data. Only the relevant portions of the local schemas are returned to the query origin node. Each QRF query was submitted at all leaf nodes. The average communications cost for each query is shown in Figure 15. Again, note that the data points are independent, and the curve is only used to illustrate them. Although there is no direct comparison with queries in previous cases, it is instructive to note that the scale of the QRF results is consistent with imprecise-query processing costs in those cases. In other words, the performance of a QRF query is roughly comparable to the performance of an imprecise query. The difference in costs is due to the varying number of possible semantic matches that had to be searched for each QRF query. For example, the first four queries dealt with variations of "requirements" and "personnel." There were many local databases dealing with these subjects, so there were many hypernyms chains

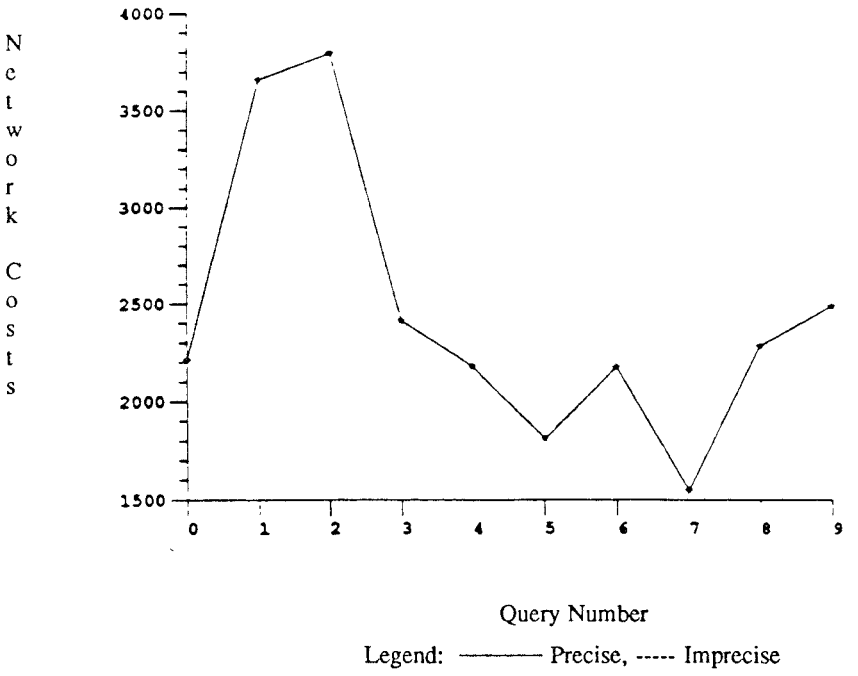


Fig. 15. Communication costs (in microseconds) for QRF queries—Case 4.

and local schemas to search (search costs were high). However, Query 7 was a search for terms semantically close to "weapon." Only a few local databases had similar terms, so the search cost was low (there were few hypernym chains that matched).

In summary, a designer or user who wants to use the QRF does not have to pay a significant performance penalty. For queries that may require multiple sources to be integrated in the result, it is reasonable to use the QRF to identify the appropriate sources. In such a case, the user would submit a QRF query, analyze the result, choose the appropriate data references, and submit a precise query with those references. The system performance would be roughly comparable to submitting two sequential queries.

6. CONCLUSIONS AND FUTURE DIRECTIONS

Multidatabase systems provide globally integrated access to multiple, autonomous local databases in a distributed system. Identification of semantically similar data across different local databases despite different data representations and naming conventions was presented as a significant problem in current research. A number of ideas from linguistic research and large-system user interface techniques were applied to develop the Summary Schemas Model (SSM) as a suitable solution to this problem.

6.1 Summary Schemas Model

The SSM was proposed as an extension of multidatabase language systems that aids in semantic identification of similar data and automatic interpretation of imprecise data references. The SSM maintains a system taxonomy that provides linguistic relationships among terms from a general English lexicon. The summary schemas hierarchy provides a concise, abstract view of the information available in the multidatabase system. Summary schemas are generated using the linguistic knowledge in the system taxonomy. The Semantic-Distance Metric (SDM) provides a quantitative measure of the semantic similarity between two terms. The SDM and the summary schemas hierarchy are used to match imprecise data references to semantically similar system access terms.

Imprecise queries allow users to specify data references in their own terms, rather than the system's terms. No current multidatabase system allows users to submit imprecise queries. The ability to automatically map imprecise user terms to precise system terms is a significant enhancement to existing multidatabase system interfaces. Imprecise-query processing provides a single match to a user's imprecise data reference. The Query Refinement Facility (QRF) returns a list of matches for a user's imprecise data reference. The QRF is useful when the requested information is partitioned and/or replicated at multiple local databases. Users (particularly database designers) can use the QRF to identify all available system data that is semantically similar to their request.

6.2 Imprecise Query-Processing Performance Evaluation

A simulator was developed to study the overhead costs of imprecise-query processing. Communications and processing costs were measured for precise queries in a simulated multidatabase language system and compared to the costs for semantically equivalent imprecise queries. The types of queries, distribution of local-database schemas, network topology, node-processing speeds, and result size were varied for different experiments. Best- and worst-case scenarios for imprecise-query processing were studied. A variety of sample queries and sample database schemas was used to study average-case scenarios.

It was shown that imprecise queries can actually provide better performance (lower communications and/or processing costs) than corresponding precise queries if imprecise-query processing identifies a semantic match that is closer (in the network) to the query origin node than the precise data reference. Imprecise queries can also perform better than precise queries if imprecise-query processing locates a semantic match at a smaller database (than the precise reference). A smaller database can be accessed faster and tends to produce smaller results (lesser result communications cost). However, in most cases, imprecise-query processing added some overhead costs, relative to precise-query processing, because of the additional message passing and processing required by imprecise-reference resolution. For large results, this overhead was negligible because the costs of result processing

dominated the measurements. In most average scenarios, imprecise-query–processing overhead was relatively small. QRF processing has no corresponding function in standard multidatabase systems, so there was no basis for comparison of the QRF measurements. However, it was shown that QRF processing costs were roughly comparable to the costs for imprecise queries. QRF does not actually make any data accesses (at the local databases), but does perform a more extensive search of the summary schemas hierarchy.

6.3 Future Research

A number of areas have been identified for further work on the SSM. A key area is the need to develop a more sophisticated system taxonomy. Roget provides only the most basic semantic relationships, and the vocabulary is somewhat dated. In particular, Roget's hypernym terms are not as meaningful and intuitive as they could be for SSM purposes. The Lexical Systems Group at IBM Research has reported on promising work for a more suitable taxonomy.

SDM sophistication is currently limited by the available system taxonomy. A more complex taxonomy (with more linguistic information) would allow more variation and control over SDM calculations. The SDM is the core function that applies the power of the system taxonomy to provide semantically meaningful results to users. More sophisticated SDM equations would allow more experimentation on defining imprecision and semantic identification with meaningful (to the user) results.

Imprecise-query processing automatically provides a single semantically meaningful result to an imprecise user request. However, as information becomes more fragmented and user requests become more complex, the system will need increasing capability to automatically identify multiple semantic matches which must be integrated. The current QRF is a step in that direction. More research remains to be done in the area of interpreting user requests that span multiple data sources.

Finally, the SSM is intended to be a useful tool for operational use. The SSM simulator has shown that imprecise-query–processing performance is competitive with (existing) precise-query–processing results. Moreover, imprecise-query processing is a user function unavailable on current multidatabase systems. Ongoing work should concentrate on actually implementing the SSM.

The Summary Schemas Model provides a significant enhancement to existing multidatabase system user interfaces by allowing users to specify queries in their own terms. It has a solid basis in existing linguistic theory and has been shown to provide reasonable performance when compared with currently available functions (where they are comparable).

REFERENCES

AITCHISON, J., AND GILCHRIST, A. 1987. *Thesaurus Construction.* 2nd ed. Aslib, London

ANDREW, D. 1987. INGRES/STAR. A product and application overview. In *Colloquium on Distributed Database Systems.* IEE, London, 21–27.

BATINI, C., LENZERINI, M., AND MOSCARINI, M. 1983. Views integration. In *Methodology and Tools for Data Base Design*. North-Holland, Amsterdam, 57–84.

BATINI, C., LENZERINI, M., AND NAVATHE, S. B. 1986. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv. 18*, 4 (Dec.), 322–364.

BELCASTRO, V., DUTKOWSKI, A., KAMINSKI, W., KOWALEWSKI, M., MALLAMICI, C. L., MEZYK, S., MOSTARDI, T., SCROCCO, F. P., STANISZKIS, W., AND TURCO, G. 1988. An overview of the distributed query system DQS. In *Proceedings of the International Conference on Extending Database Technology-EDBT '88*. Springer-Verlag, Berlin, 170–89.

BELL, D. A., GRIMSON, J. B., AND LING, D. H. O. 1987. EDDS—a system to harmonize access to heterogeneous databases on distributed micros and mainframes. *Inf. Softw. Tech. 29*, 7 (Sept.), 362–370.

BREITBART, Y. J., AND TIEMAN, L. R. 1984. ADDS—heterogeneous distributed database system. In *Proceedings of the 3rd International Seminar on Distributed Data Sharing Systems*. North-Holland, Amsterdam, 7–24.

BREITBART, Y, GARICA-MOLINA H. AND SILBERSCHATZ, A. 1992. Overview of multidatabase transaction management. *VLDB J.* 1, 2, 181–240.

BRIGHT, M. W., AND HURSON, A. R. 1991. Linguistic support for semantic identification and interpretation in multidatabases. In *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*. IEEE Computer Society Press, Los Alamitos, Calif., 306–313.

BRIGHT, M. W., AND HURSON, A. R. 1990. Summary schemas in multidatabase systems. Computer Engineering Tech. Rep. TR-90-076. The Pennsylvania State Univ., University Park, Penn.

BRIGHT, M. W., HURSON, A. R., AND PAKZAD, S. H. 1992. A taxonomy and current issues in multidatabase systems. *IEEE Comput. 25*, 3 (Mar.), 50–60.

BYRD, R. J. 1989. Discovering relationships among word senses. In *Dictionaries in the Electronic Age—Proceedings of the 5th Annual Conference at the Centre for the New Oxford English Dictionary*. University of Waterloo, Waterloo, Canada.

CERI, S., AND PELAGATTI, G. 1984. *Distributed Databases: Principles and Systems*. McGraw-Hill, New York.

CERI, S., PERNICI, B., AND WIEDERHOLD, G. 1987. Distributed database design methodologies. *Proc. IEEE 75*, 5 (May), 533–546.

CHODOROW, M. S., RAVIN, Y., AND SACHAR, H. E. 1988. A tool for investigating the synonymy relation in a sense disambiguated thesaurus. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*. 144–151.

COLLIER, G. H., AND FELLBAUM, C. 1988. Exploring the verb lexicon with the sensus electronic thesaurus. In *Proceedings of the 4th Annual Conference of the Centre for the New Oxford English Dictionary: Information in Text*. University of Waterloo, Waterloo, Canada, 11–20.

D'ATRI, A., AND TARANTINO, L. 1989. From browsing to querying. *Data Eng. 12*, 2 (June), 46–53.

DAYAL, U., AND HWANG, H. 1984. View definition and generalization for database integration in a multidatabase system. *IEEE Trans. Softw. Eng. 10*, 6 (Nov.), 628–644.

DEEN, S. M., AMIN, R. R., AND TAYLOR, M. C. 1987. Implementation of a prototype for PRECI*. *Comput. J. 30*, 2 (Feb.), 157–162.

DUTCH, R. A., ED. 1965. *Roget's Thesaurus*. St. Martin's Press, New York.

FANKHOUSER, P., LITWIN, W., NEUHOLD, E. J., AND SCHREFL, M. 1988. Global view definition and multidatabase languages—two approaches to database integration. In *Proceedings of the European Teleinformatics Conference—EUTECO '88 on Research into Networks and Distributed Applications*. North-Holland, Amsterdam, 1069–1082.

FUHR, N. 1990. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of the 16th International Conference on Very Large Data Bases*. Morgan Kaufman Publishers, Los Altos, Calif., 696–707.

FURNAS, G. W., LANDAUER, I. K., GOMEZ, L. M., AND DUMAIS, S. T. 1987. The vocabulary problem in human-system communication. *Commun. ACM 30*, 11 (Nov.), 964–971.

GOVE, P. B., ED. 1963. *Webster's Seventh New Collegiate Dictionary*. G.& C. Merriam, Springfield, Mass.

HURSON, A. R., AND BRIGHT, M. W. 1991a. Global information access for microcomputers. *J. Mini Micro Comput. Appl. 10*, 2, 73–81.

HURSON, A. R., AND BRIGHT, M. W. 1991b. Multidatabase systems: An advanced concept in handling distributed data. In *Advances in Computers*, vol. 32. Academic Press, San Diego, Calif., 217–268.

HURSON, A. R., MILLER, L. L., PAKZAD, S. H., AND CHENG, J. B. 1990. Specialized parallel architectures for textual databases. In *Advances in Computers*, vol. 30. Academic Press, San Diego, Calif., 1–37.

HURSON, A. R., MILLER, L. L., PAKZAD, S. H., EICH, M. H., AND SHIRAZI, B. 1989. Parallel architectures for database systems. In *Advances in Computers*, vol. 28. Academic Press, San Diego, Calif., 108–151.

JAKOBSON, G., PIATESKY-SHAPIRO, G., LAFOND, C., RAJINIKANTH, M., AND HERNANDEZ, J. 1988. CALIDA: A system for integrated retrieval from multiple heterogeneous databases. In *Proceedings of the 3rd International Conference on Data and Knowledge Bases*. Morgan Kaufman, San Mateo, Calif., 3–18.

KLAVANS, J., BYRD, R., WACHOLDER, N., AND CHODOROW, M. 1991a. Taxonomy and polysemy. IBM Res. Rep. RC 16443, IBM Corporation, Yorktown Heights, N Y

KLAVANS, J., CHODOROW, M. S., AND WACHOLDER, N. 1991b. From dictionary to knowledge base via taxonomy. IBM Res. Rep. RC 16464, IBM Corporation, Yorktown Heights, N.Y.

LANCASTER, F. W. 1986. *Vocabulary Control for Information Reetrieval*. 2nd ed. Information Resources Press, Arlington, Va.

LESK, M. 1989. What to do when there's too much information. In *Hypertext '89 Proceedings*. ACM Press, New York. 305–318.

LITWIN, W. 1988. From database systems to multidatabase systems: Why and how. In *Proceedings of the 6th British National Conference on Databases*. Cambridge University Press, Cambridge, England, 161–188.

LITWIN, W. 1985a. Implicit joins in the multidatabase system MRDSM. In *Proceedings of the 9th International Computer Software and Applications Conference, COMPSAC 85*. IEEE Computer Society, Washington, D.C., 495–504.

LITWIN, W. 1985b. An overview of the multidatabase system MRDSM. In *Proceedings of the ACM Annual Conference*. ACM Press, New York, 524–533.

LITWIN, W. 1984. Concepts for multidatabase manipulation languages. In *Proceedings of the 4th Jerusalem Conference on Information Technology*. IEEE Computer Society, Washington, D.C., 309–317.

LITWIN, W., AND ABDELLATIF, A. 1987. An overview of the multidatabase manipulation language MDSL. *Proc. IEEE 75*, 5 (May), 621–632.

LITWIN, W., AND ABDELLATIF, A. 1986. Multidatabase interoperability. *IEEE Comput. 19*, 12 (Dec.), 10–18.

LITWIN, W., AND VIGIER, P. 1986 Dynamic attributes in the multidabase system MRDSM. In *Proceedings of the 2nd International Conference on Data Engineering*. IEEE Computer Society, Washington, D.C., 103–110.

LITWIN, W., AND ZEROUAL, A. 1988. Advances in multidatabase systems. In *Proceedings of the European Teleinformatics Conference—EUTECO '88 on Research into Networks and Distributed Applications*. North-Holland, Amsterdam, 1137–1151.

MOTRO, A. 1990. Accommodating imprecision in database systems: Issues and solutions. *SIGMOD Rec. 19*, 4 (Dec.), 69–74.

MOTRO, A. 1989. A trio of database user interfaces for handling vague retrieval requests. *Data Eng. 12*, 2 (June), 54–63.

NELSON, S. J., TUTTLE, M. S., COLE, W. G., SHERETZ, D. D., SPERZEL, W. D., ERLBAUM, M. S., FULLER, L. L., AND OLSON, N. E. 1991. From meaning to term: Semantic locality in the UMLS metathesaurus. In *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*. McGraw-Hill, New York, 209–213

RAFII, A., AHMED, R., DESMEDT, P., KENT, B., KETABCHI, M., LITWIN, W, AND SHAN, M. 1991. Multidatabase management in pegasus. In *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*. IEEE Computer Society Press, Los Alamitos, Calif., 166–173.

RAVIN, Y. 1990. Disambiguating and interpreting verb definitions. IBM Res. Rep. RC 15655, IBM Corporation, Yorktown Heights, N.Y.

RAVIN, Y. 1989. Synonymy from a computational point of view. IBM Res. Rep. RC 14962, IBM Corporation, Yorktown Heights, N.Y.

RAVIN, Y., CHODOROW, M. S., AND SACHAR, H. E. 1988. Tools for lexicographers revising an online thesaurus. IBM Res. Rep. RC 13903, IBM Corporation, Yorktown Heights, N.Y.

SARMA, H. 1992. The summary schema model for multidatabase systems: A simulation. Master's thesis, Dept. of Electrical and Computer Engineering, The Pennsylvania State Univ., University Park, Penn.

SHETH, A. P., AND LARSON, J. A. 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv. 22*, 3 (Sept.), 183–236.

STANISZKIS, W. 1986. Integrating heterogeneous databases. CRAI State of the Art Report P229-47, Pergamon Infotech, Maidenhead Berkshire, England.

STRONG, G. W., AND DROTT, M. C. 1986. A thesaurus for end-user indexing and retrieval. *Inf. Proc. Manage. 22*, 6, 487–492.

TEMPLETON, M., BRILL, D., DAO, S. K., LUND, E., WARD, P., CHEN, A. L. P., AND MACGREGOR, R. 1987. Mermaid—a front-end to distributed heterogeneous databases. *Proc. IEEE 75*, 5 (May), 695–708.

THOMAS, G., THOMPSON, G. R., CHUNG, C. W., BARKMEYER, E., CARTER, F., TEMPLETON, M., FOX, S., AND HARTMAN, B. 1990. Heterogeneous distributed database systems for production use. *ACM Comput. Surv. 22*, 3 (Sept.), 237–266.

WANG, Y. R., AND MADNICK, S. E. 1989. The inter-database instance identification problem in integrating autonomous systems. In *Proceedings of the 5th International Conference on Data Engineering*. IEEE Computer Society, Washington, D.C., 46–55.

YU, C., SUN, W., DAO, S., AND KEIRSEY, D. 1991. Determining relationships among attributes for interoperability of multi-database systems. In *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*. IEEE Computer Society Press, Los Alamitos, Calif., 251–257.

Received May 1992; revised September 1993