KADBASE -- A Prototype Expert System-Database Interface for Integrated CAE Environments

H. Craig Howard
Assistant Professor
Department of Civil Engineering
Stanford University
Stanford, CA 94305

Daniel R. Rehak
Senior Scientist
Expert Technologies, Inc.
Pittsburgh, PA 15213

Abstract

Database management systems (DBMSs) are important components of existing integrated computer-aided engineering (CAE) systems. Expert systems (ESs) are being applied to a broad range of engineering problems. However, most of the prototype expert system applications have been restricted to limited amounts of data and have no facility for sophisticated data management. KADBASE is a flexible, knowledge-based interface in which multiple expert systems and multiple databases can communicate as independent, self-descriptive components within an integrated, distributed engineering computing environment.

I. INTRODUCTION

Integrated engineering computing systems have evolved into sets of algorithmic programs that revolve around a central database management system (DBMS). The DBMS frees the application subsystems from the details of managing data storage and retrieval while providing a common pool of information that cooperating subsystems can share. Now the character of integrated engineering systems is changing. Knowledge-based programming techniques, specifically expert systems (ES), are being applied to a broad range of engineering problems. However, most of the prototype expert system applications have been restricted to limited amounts of data and have no facility for sophisticated data management. As expert systems are integrated into engineering computing environments, the data management capabilities of the integrated systems must be adapted to serve these new components. Likewise, expert systems must evolve to incorporate capabilities to access large shared databases.

Integrated systems are built from a combination of individual programs (both algorithmic and knowledge-based). Each of these has its own data structures, databases and information models. Integrating these disparate data models into a single central common database is complex and, in many cases, unrealistic or undesirable. An alternative approach is to develop an integrated system which recognizes this disparity and is designed to deal with multiple databases. Such a collection of databases is likely to be quite heterogeneous, i.e., a variety of DBMSs with different data models, varying implementations, and operating on different hardware. Several systems have been proposed to support networks of heterogeneous database management systems [Adiba 78, Cardenas 80, Smith 81, Jakobson 86]. These DBMS networks provide users with access to multiple databases on a computer network while hiding the details of network communications and allowing the users (or application programs) to treat data within the contexts of their own data representations. These techniques can be applied the design of an engineering expert system-database interface by adapting the model to accommodate the data access needs of expert systems and extending the data representation capabilities to account for the complexities of engineering data.

KADBASE [Howard 86a][1] is a prototype of a flexible, knowledge-based interface in which multiple expert systems, or more generally knowledge-based systems (KBSs), and multiple databases can communicate as independent, self-descriptive components within an integrated, distributed engineering computing system. The interface takes a data request from a expert system, performs the indicated operations using the available DBMSs, and returns a reply to the expert system. Each expert system and database is linked only to the interface; therefore, new ESs and DBMSs can be added to the integrated environment with ease. KADBASE can be generalized to serve all the components of the engineering application, both algorithmic and non-algorithmic, providing the basis for a large-scale integrated engineering environment composed of diverse software systems running on heterogeneous hardware.

II. KADBASE ARCHITECTURE

KADBASE is a prototype distributed network database interface between database management systems and knowledge-based system components of an integrated CAE system. The interface processor responds to data requests by using the declarative knowledge about the data spaces of the components being interfaced in conjunction with its own general knowledge about processing requests and interpreting the components' data descriptions. Because the information required for reasoning about each component is represented separately as descriptive knowledge, the interface is more flexible than purely algorithmic linkages in which the descriptive information is embedded in the processing instructions. Furthermore, each expert system and database is linked only to the interface; therefore, new

[1]The issues and motivation behind the development of KADBASE have been explored in earlier papers [Rehak 85, Howard 85, Howard 86b].

ESs and DBMSs can be added to the integrated environment with ease.

The multiple DBMS networks mentioned earlier provide a conceptual model for the organization of KADBASE. The information contained in the schemata of the individual engineering databases is integrated into a single *global schema* that is based on a semantic database model. A request for data issued by a knowledge-based system (KBS) is translated (mapped) from the data manipulation language (syntax) and data structure (semantics) of the requesting component into a global syntax referencing the global schema. The mapping process is formally divided into two separate processes: first, a *syntactic translation* from the KBS data manipulation language to the global data manipulation language; and second, a *semantic* translation from the KBS data structure to the global data structure. After the request is mapped, the interface processor identifies a set of target databases that contain information required to answer the query and generates *subqueries* to those databases to gather that information. Each subquery to a target database is translated to the specific database syntax and semantics, and the corresponding database manager is invoked to process the resultant subquery. Inverse mappings return the results to the requesting component.

KADBASE is divided into three basic components as described below.

- The **Knowledge-Based System Interface** (KBSI) is a part of each knowledge-based system. It formulates the queries and updates sent to the network data access manager and processes the replies from the network data access manager. The KBSI possesses knowledge about the schema of the KBS context (data space) and uses that knowledge to perform semantic (and syntactic translations) for the queries, updates and replies.
- The **Knowledge-Based Database Interface** (KBDBI) acts as an intelligent front-end for a "standard" DBMS. It accepts queries and updates from the network data access manager and returns the appropriate replies. Like the KBSI, the KBDBI possesses knowledge about the local database schema and the local language for data manipulation requests. It uses that knowledge to perform semantic and syntactic translations for the queries, updates and replies.
- The **Network Data Access Manager** (NDAM) provides the actual interface. It receives requests (queries and updates) expressed in terms of the global schema from the knowledge-based systems (through their KBSIs). Using information associated with the global schema, the NDAM locates sources for the data referenced in a request and decomposes each request into a set of subqueries or updates to the individual *target* databases. The subrequests are sent to the corresponding knowledge-based database interfaces (KBDBIs) for processing. The replies from the KBDBIs are combined to form a single reply to the original request and sent to the requesting application through its KBSI.

In KADBASE, components are organized into knowledge-based systems, with knowledge grouped into *knowledge modules* (KMs) (processing knowledge about particular subproblems) and *knowledge sources* (KSs) (passive, descriptive information about the knowledge-based component). KMs typically perform control and translation tasks, while KSs are used to represent schema descriptions.

Throughout KADBASE, a frame data model is used to represent various types of knowledge including schema definitions, syntactic translation procedures, queries and replies. As used in KADBASE, the frame data model uses *frames* to represent objects. A frame consists of *slots* that contain values describing the object. Slots may be *attributes*, which contain simple descriptive values, or *relationships*, which serve to link the object to another type of object to provide *inheritance*. In the remainder of this paper, frame names are typeset in **bold face**, and slot names are typeset in SMALL CAPITALS.

The remainder of this section presents an overview of the knowledge representation and translation processing in KADBASE. The first subsection discusses the syntactic translation process. The next two subsections describe the organization of the schema description knowledge and the semantic translation process.

A. Syntactic Mapping

The syntactic translation in KADBASE is the transformation of requests (queries and updates) between the external data manipulation languages (e.g., QUEL, SQL) and the internal KADBASE request representation. In KADBASE, the local system (KBSI or KBDBI) is responsible for performing the syntactic translation. The syntactic translation is dependent only on the local data manipulation language; the same syntactic processors may be used for multiple applications of the same database management system or expert system building tool. Each syntactic processor maps requests between two fixed language: the component data manipulation language and the KADBASE internal representation. Since the translation task is does not vary with the applications, the syntactic processor may be implemented as a special-purpose program using an algorithmic approach.

Internally, KADBASE uses networks of frames to represent requests. The organization of the request frame representation serves as the global data manipulation language, paralleling the global schema. The precise character of the internal request representation is not important to the discussion of remainder of the translation process and is therefore omitted from this paper.

B. Schema Description and Mapping Knowledge

KADBASE integrates the components' data spaces through a global schema based on frame data model. The basic unit of the frame model used in KADBASE is the *entity*, represented by a frame. An entity is "*any distinguishable object* -- where the 'object' in question may be as concrete or as abstract as we please" [Date 83].

The schema description knowledge required by the interface is partitioned into three levels: the component's local data representation expressed in the component's own data model (hierarchical, network, relational, frame-based, object-oriented, etc.), the component's local data representation expressed in terms of the global frame model, and the global schema expressed in terms of the global frame model. This information is represented as *knowledge sources* within the knowledge-based components of KADBASE. The three types of schema description knowledge sources are described more fully below:

- The **Local Schema** (LS) describes the organization of a component's local data structure in terms of the local data model. The character of the local schema is highly dependent on the type of database management system (DBMS) or KBS being described. In a relational DBMS, the local schema consists of the definitions of the relations and their attributes in the database. In a object-oriented expert system, the local schema consists of the definitions of the hierarchy of object classes in the context.
- The **Local Frame-Based Schema** (LFBS) represents the local schema in the semantics of the frame data model. The LFBS should be a fully updatable view of the underlying data structure expressed in the local terminology (the names for entities and attributes). The organization of the LFBS may differ that of the local schema because the LFBS may group slots from several underlying data structures (relations, frames, etc) into a single entity if those data structures have the same primary or candidate keys. The LFBS should also contain or be capable of referencing all the information in the local schema with respect to constraints, key attributes, and domain properties (data types, ranges, dimensions, etc.). The LFBS may contain information about semantic relationships between entities not found in the underlying data models; thus, the LFBS may be used to provide an enhanced semantic data model when the local model lacks the capabilities to express relationships between entities.
- The **Global Schema** (GS) represents the common data space shared by all components in the integrated system. In effect, it is the union of the frames and slots in the LFBSs. Since LFBSs may differ with respect to terminology (the names for common frames and slots) and slot domains (data types and dimensions), the establishment of the global schema involves the selection of a single set of global names and domains. This selection is performed by a global schema administrator, who is responsible for the consistency and completeness of the global schema.

The three schema description knowledge sources each serve to represent a specific data structure in terms of a specific data model. They do not contain knowledge about how to relate that data structure to the other schema representations, i.e., how to map between schemata (e.g., LS to LFBS and LFBS to GS). That schema mapping knowledge is contained in two additional knowledge sources:

- The **Local Frame-Based Mapping** (LFBM) describes the organizational mapping between the local schema model (LS) and the local frame-based schema (LFBS). The LFBM is necessary because the LFBS may group slots from several underlying LS data structures (relations, frames, etc) into a single entity if those data structures have the same primary or candidate keys. It contains the information relating each slot in the LFBS with its counterparts in the LS.
- The **Local Integration Mapping** (LIM) contains mapping information necessary to integrate the LFBS into the global schema (GS). The LIM is necessary because the LFBS can differ from the global schema in two ways: in the names for entities and slots, and in the domains of the attribute values. The LIM represents the former with a set of terminology (name) mappings for the entity and slot names, and the latter with domain mappings for attribute values (tables or functions that map local values into the corresponding global values when the attribute domains are different). Domain mappings are required whenever the local component represents the value of an attribute in different terms than the global schema. For instance, the global schema may represent the value of an enumerated type with a descriptive string (e.g., "3000 psi concrete") while a database may store that value as an integer code (e.g., 34).

Two additional mapping knowledge sources are required at the global level.

- The **Global Data Source Mapping** (GDSM) relates each slot (data item) in the global schema to the list of databases and KBS contexts in which the slot can be found.
- The **Global Integration Mapping** (GIM) contains constraints and functions relating the slots in the global schema. The mappings represent mathematical and logical interrelationships between attributes found in different databases and KBS contexts. A constraint such as "AREA = WIDTH * BREADTH" may represent a mapping

between one database that describes rectangular entities by area alone and another that uses width and breadth instead.

In the KADBASE prototype, the integration of the local frame-based schemata into the global schemata is performed manually. The task of defining a global view and relating it to each local view of data requires substantial domain knowledge and intelligent reasoning. The implementation of an intelligent schema integrator was not attempted in this project. Therefore, KADBASE requires that a global database administrator standardize the terminology for the global entities and slots, select global slot domain properties (data types and units), and define global relationships and constraints.

C. Semantic Mapping

The semantic translation of requests (queries and updates) and replies (data) in KADBASE is independent of the types of KBS and DBMS components involved. Semantic processing is based on the information provided in the schema description and mapping knowledge sources described in the previous section. Therefore, the semantic translation is performed by an application-independent knowledge module that may be invoked by any component.

The semantic translation process can be divided into two steps: local schema (LS) to local frame-based schema (LFBS) and local frame-based schema to global schema (GS). The first step involves only local information (LS, LFBM, and LFBS) and, therefore, should be performed within the knowledge-based system interface or knowledge-based database interface using the application-independent semantic translation module. The second step involves both local and global information (LFBS, LIM, and GS) and maybe implemented either locally or globally.[2]

In addition to the two steps described above, the semantic translation process can be divided into two types (requests and updates) and two directions (local to global and global to local). For convenience, these divisions can be grouped according to the flow of requests and replies through the system as follows:

- ***Request Translation***
  - **LS (KBS) to LFBS** -- The only differences between the LS and LFBS that affect requests are organizational differences; i.e., the attributes of the entities may be distributed in separate data structures in the LS (e.g., the single entity *beam* may be represented by two objects in the context: *beam-location* and *beam-type*). These organizational differences are represented by knowledge in the LFBM. Semantic translation at this level consists of changing references to entities and slots from their designations in the LS to their designations in the LFBS and removing clauses from the qualifier that denote links between local data structures that are represented as a single entity in the LFBS.
  - **LFBS to GS** -- The semantic mapping process between the LFBS and the GS involves name changes for the entities and slots as well as domain mappings for the slot values. These mappings are represented in the LIM. The name changes in the LFBS to GS translation are trivial, one-to-one mappings. These name changes are required wherever a slot or entity reference appears in the request. Domain mapping may involve data type conversions (e.g., real-to-integer), unit conversions (e.g., inches-to-feet), tabular mappings (representing one-to-one correspondences between local and global domain values) or functional mapping (non-one-to-one correspondences). Data type and unit conversions are implemented by replacing the slot reference with a mathematical expression representing the mapped value (e.g., *"fix(real)"* for real-to-integer, *"length/12"* for inches-to-feet, etc.). Tabular and functional domain mappings are implemented only for qualifier expressions of the form "<slot with domain mapping> <comparison operator> <value>"; in those expressions, the "<value>" is mapped from the local domain into the global domain (e.g., *"color = 6"* is mapped to *"color = 'indigo'"*).
  - **GS to LFBS** -- The semantic translations for name and domain mappings from the GS to the LFBS is basically the same as the LFBS to GS mapping described above, only implemented in reverse.
  - **LFBS to LS (DBMS)** -- The only differences between the LS and LFBS that affect requests are organizational differences as described previously. Once again, these organizational differences are represented by knowledge in the LFBM. The semantic translation at this level consists of changing references to entities and slots from their designations in the LFBS to correspond to their designations in the LS and adding clauses to the qualifier to denote links between local data structures that are represented as a single entity in the LFBS.
- ***Reply Translation*** (Now the object of the translation is a set of data.)
  - **LS (DBMS) to LFBS** -- Since the map-

[2]In the prototype implementation, LFBS to GS translation is performed at the local level.

ping between the LS and the LFBS is purely one of organization and naming, no reply translation is required.

- **LFBS to GS** -- For replies, only domain mappings are required in the translation between the LFBS and the GS. The types of domain mappings are the same as for request mappings (data type, unit, tabular, and functional mappings), but for replies the conversions are applied directly to the data being returned.
- **GS to LFBS** -- As above, the required translations between the GS and LFBS consist of domain mappings applied directly to the data.
- **LFBS to LS (KBS)** -- Since the only differences between the LFBS and the LS are in terms of attribute grouping, no semantic translation of replies is required at this level.

III. Implementation and Applications

KADBASE provides the mechanism to develop a distributed, integrated engineering computing system composed of the components described above. Communication between components is isolated in a communications module associated with each component. This module hides the physical message passing mechanism. Thus, the distributed nature of the KADBASE architecture is hidden from the user and applications (databases and knowledge-based systems may co-exist on a single machine or on multiple, heterogeneous machines). In a distributed environment, each component may be implemented as a separate process. In that case, the NDAM and KBDBIs function as *servers* responding to incoming requests.

The KADBASE prototype has been test in conjunction with two knowledge-based structural engineering application systems:

- **SPEX** (Standards Processing Expert) [Garrett 86] -- is a knowledge-based, structural component design system developed by James Garrett. SPEX uses KADBASE to provide access to a database of standard structural steel members for use in its component design process.
- **HICOST** -- a knowledge-based cost estimator for detailed building designs developed to demonstrate KADBASE. HICOST uses KADBASE to access a multiple databases (a building design database, a project management database, and a library database of unit costs).

KADBASE and its demonstration applications are implemented in a distributed computing environment consisting of a DEC VAX 11/750 and several MicroVAXs using the Mach operating system and linked by Ethernet. Franz Lisp [Foderaro 82] is the principal programming language used in implementation. The KADBASE sample databases are supported by the INGRES database management system [Stonebraker 76].

References

[Adiba 78] Adiba, M., and Portal, D., "A Cooperation System for Heterogeneous Data Base Management Systems," *Information Systems*, Vol. 3, No. 3, pp. 209-215, 1978.

[Cardenas 80] Cardenas, A., and Pirahesh, M. H., "Data Base Communication in a Heterogenous Data Base Management System Network," *Information Systems*, Vol. 5, No. 1, pp. 55-79, 1980.

[Date 83] Date, C. J., *An Introduction to Database Systems*, Vol. II, *The System Programming Series*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1983.

[Foderaro 82] Foderaro, J. K., and Skower, K. L., *The FRANZ LISP Manual*, University of California at Berkeley, 1982.

[Garrett 86] Garrett, J.H., *SPEX -- A Knowledge-Based Standards Processor for Structural Component Design*, unpublished Ph.D. Dissertation, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, September 1986.

[Howard 85] Howard, H.C., and Rehak, D.R., "Knowledge Based Database Management for Expert Systems," *ACM Sigart Newsletter*, Special Interest Group on Artificial Intelligence, Association for Computing Machinery, Spring 1985.

[Howard 86a] Howard, H. C., and Rehak, D. R., *Interfacing Databases and Knowledge Based Systems for Structural Engineering Applications*, Technical Report EDRC-12-06-86, Engineering Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, November 1986.

[Howard 86b] Howard, H.C., and Rehak, D.R., "Expert Systems and CAD Databases," *Knowledge Engineering and Computer Modelling in CAD*, CAD86: Seventh International Conference on the Computer as a Design Tool, London, pp. 236-248, September, 1986.

[Jakobson 86] Jakobson, G., Lafond, C., Nyberg, E., and Piatetsky-Shapiro, G., "An Intelligent Database Assistant," *IEEE Expert*, Vol. 1, No. 2, pp. 65-78, Summer 1986.

[Rehak 85] Rehak, D.R., and Howard, H.C., "Interfacing Expert Systems with Design Databases in Integrated CAD Systems," *Computer-Aided Design*, November 1985.

[Smith 81] Smith, J. M., et al., "Multibase -- Integrating Heterogenous Distributed Database Systems," *AFIPS Conference Proceedings*, Vol. 50, pp. 487-499, 1981.

[Stonebraker 76] Stonebraker, M., Wong, E., and Kreps, P., "The Design and Implementation of INGRES," *ACM Transactions on Database Systems*, Vol. 1, No. 3, pp. 189-222, September 1976.