Extending the Data Dictionary for Data/Knowledge Management

Cecile L. Hydrick
and
Dr. Sara J. Graves
Computer Science Department
University of Alabama in Huntsville

Abstract

Current relational database technology provides the means for effeciently storing and retrieving large amounts of data. By combining techniques learned from the field of artificial intelligence with this technology, it is possible to expand the capabilites of such systems. This paper suggests using the expanded domain concept, an object-oriented organization, and the storing of knowledge rules within the relational database as a solution to the unique problems associated with CAD/CAM and engineering data.

1. Introduction

Data management for NASA often involves large amounts of diverse data stored on many different computer systems and at many different geographical locations. Types of data which must be tracked include project management data, financial and budgetary data, CAD/CAM data, engineering data, and documents. The possibility of using a single relational database management system (DBMS) for data connectivity has been explored. However, CAD/CAM and engineering data present problems which are not being currently addressed by existing DBMS products.

Although CAD/CAM and engineering data have the same basic requirements for storage and retrieval, certain characteristics of the data show why existing DBMS's fail. Such data (1) tends to be heterogeneous, consisting of graphical, textual, procedural, and mathematical data; (2) requires a dynamic schema as entities are created and destroyed; (3) tends to be object-oriented with complex relationships associated with the objects; and (4) exhibits object-specific relationships which change from object to object [5].

These characteristics require that a database designed for such applications (1) be able to represent a wide range of data types, (2) be able to represent complex relationships between data items, (3) and be able to represent certain "knowledge" about that data [2]. Existing commercial DBMS's do not at the present time have those capabilities.

The traditional data dictionary/directory may provide the answer. The data dictionary contains the "meta-data" which is the description of the data in the database. By extending the descriptions using knowledge representation techniques from the field of artificial intelligence (AI), the dictionary can in effect become the "knowledge base" for the DBMS, providing both dynamic schema generation and extended data types.

This paper presents methods for extending the data dictionary in a relational database management system by extending the domain concept to allow the representation of many different data types. Using an object-oriented model allows the expression of complex relationships between objects. "Knowledge" can be stored in the form of production rules mapped into a relational table. Combining the extension of the domain, the object-oriented model, and the storage of production rules in relational tables produces a data dictionary that is dynamic and capable of evolving over time, thus meeting the needs of CAD/CAM and engineering database applications.

II. Problems with Existing Data Dictionaries

The traditional process of database development resulted in a collection of static record structures which remained fixed throughout the life of the database applications. Data base administrators typically regarded data dictionaries/directories as static tools to aid them in the control of information resources [5].

The advent of CAD/CAM and engineering database systems has created the need for data dictionary definition to occur throughout the life of the application as objects are created, modified, and destroyed. The data dictionary, if viewed as a "knowledge base" rather than a collection of static records, can play an active role in this process. Expert knowledge about database design can be stored in the data dictionary itself, thus allowing for the creation of schemas as data loading occurs.

The key concept in the above scenario is that meta-data inherently contains knowledge which can be exploited for dynamic schema generation and knowledge management purposes. However, this will require that future systems be more tightly integrated than at present. In order to take full advantage of the knowledge inherent in the meta-data, data and meta-data can no longer be functionally separated, but must be made co-resident in the same "knowledge base". In this approach, database instances, data types, operations, and transactions are viewed as "objects". Two issues must be addressed in designing a knowledge-based data dictionary: a scheme for knowledge representation and the integration of the data and meta-data [5].

III. Moving from Data Management Towards Knowledge Management

Current DBMS's are effective in storing and retrieving large amounts of data. However, while a typical data dictionary may describe the physical size of the attribute "employee number", it may have no way to represent the fact that EMPLOYEE is a subtype of PERSON [11].

Artificial Intelligence research has produced Knowledge Representation Systems (KRS) which attempt to model the way in which human knowledge is represented and acquired. However, these systems have not been able to efficiently exploit large amounts of data due to the fact that they tend to be memory-based rather than disk-based [2]. Because they also tend to have high overhead and have ignored the issues of backup and error recovery, they have not been considered practical for large commercial applications [11].

Recent research has centered around finding ways to combine the best of both the DBMS and the KRS. There have been four approaches to integrating the two systems:

(1) integrate an existing AI system with a DBMS;

(2) enhance an AI system with data functionality;

(3) tightly integrate AI and database by designing an entirely new system;

(4) extend a DBMS by enhancing the data model with knowledge representation and other AI capabilities [2].

The fourth alternative forms the basis for this paper. This approach actually involves mapping knowledge into the existing DBMS. Several techniques have been used including the assorted semantic data models and the mapping of production rules into relational tables. An integrated approach must combine modelling richness with knowledge rules for inferencing capability.

IV. Expanding Domains in the Relational Model

Wedekind [10] discusses the importance of the domain in the design of conceptual schemata. He argues that the design process should reflect a learning situation in which single elementary facts about data are combined to form more complex knowledge about the data. His approach is a constructive method in which global domains are used as a basis for building relations in a stepwise fashion.

For example, a domain description such as COLOR = {green, red,blue} might be replaced with the following elementary sentences:

COLOR is a DOMAIN;
green is a COLOR.
red is a COLOR.
blue is a COLOR;

The fact that COLOR is a domain must be established before the members of the domain can be enumerated. In the same manner, the members of the domain must be known before an entity can be described as having that attribute.

In order to implement this concept using the relational model, Wedekind suggests the addition of four relations to the metadatabase: DOMAIN, ENUM (for enumerated types), and PRE and FUN which are relations specific to INGRES which allow the use of INGRES predicates and the calling of functions for validity testing.

Expanding the domain concept in engineering and CAD/CAM databases can provide two distinct advantages: (1) abstract data typing, and (2) "built-in" integrity constraints. The ability to represent abstract data types is necessary because of the heterogeneous nature of the data, while the enumeration of domain members provides the means of checking data entities for validity.

V. Object-Oriented Organization

One method for ascribing meaning to data is to describe the data in terms of objects rather than static record structures [5]. Objects can be entities or the relationships between the entities. Both declarative and procedural information can be included in the model [9].

Objects can be organized into "semantic nets". Primitives can include:

1) The class of an object of a certain type;

2) ISA-links which relate subtypes and supertypes;

3) ASA-links stating that a token is a specific type;

4) Primitive maps and functions which provide access to meta-data [5].

Both the object-oriented model and the relational database model share the goal of logical data independence, hence a natural mapping exists between the two models [9].

An object-oriented model that can be mapped into the relational model has been described by Sheldon Borkin [1]. His model is defined as the "semantic relation data model" and is based on the premise that the current database state consists of sets of statements describing the current state of the application. These statements are built from the meanings of natural language sentences which can be expressed as a verb phrase (predicate) plus several noun phrases.

An example of how an object might be mapped to the relational model is shown below. The object in Figure 1 has been named "ARCH" and consists of three elementary items named "BEAM1", "BEAM2", and "BEAM3". The functional decomposition of this object is shown by the graph in Figure 2. The relations resulting from the graph are shown in Table 1. It is interesting to note that the relations in Table 1 look similar to the way in which PROLOG states facts - i.e., supports(object1,object2).

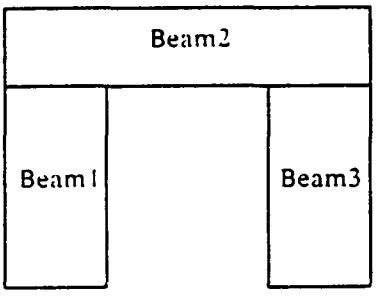

Figure 1: Object Named "Arch"

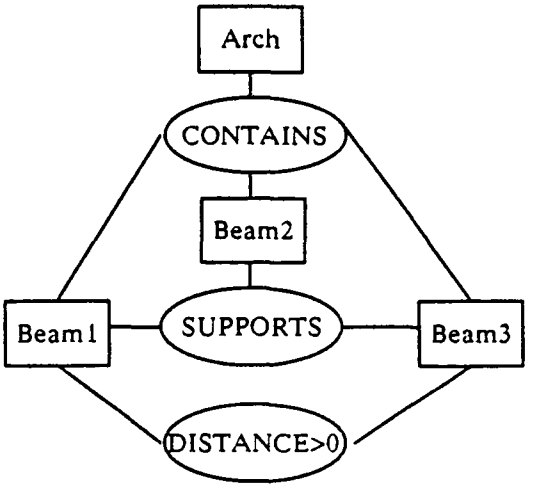

Figure 2: Functional decomposition

The object-oriented model has three advantages over the hierarchical, network, and relational models. First, the database can be viewed as a collection of abstract objects, not simply a group of two-dimensional tables. Second, both abstraction (attribute interconnection) and generalization (subtyping) can be more easily represented. Finally, object-oriented schema provide built-in integrity constraints [8]

Table 1: Relations for Object "Arch"

1) CONTAINS(Object1, Object2)

Arch1	Beam1
Arch1	Beam2
Arch1	Beam3

2) SUPPORTS(Object1, Object2)

Beam1	Beam2
Beam3	Beam2

3) DISTANCE>0(Object1, Object2)

Beam1	Beam3

As with most other existing models, trade-offs exist. Potential problems which may occur when using an object-oriented data model are added complexity and the difficulty in restructuring relationships once they have been defined [8].

VI. Storing Knowledge Rules for Inferencing

AI databases generally include two types of objects: facts about other objects and knowledge rules for inferencing [5]. In the data dictionary, facts can be represented using an object-oriented data model.

However, in order to maintain the close integration between data and meta-data, a method must be found for storing the knowledge rules within the database itself.

Recent work by Han-lin Li in China has centered on mapping production rules for inferencing into the relational model. This approach allows the relational DBMS to handle the matching and retrieval of production rules which have been mapped to the relational model.

Li's work centers around the basic form of the production rule which is "IF condition, THEN action with certainty factor CF." "AND" and "OR" operators are allowed to form complex conditions or actions. Table 2 shows some example production rules.

Table 2: Production Rules

Rule 1: If A = a and B = b and C = c
THEN: D = d1 with CF(D) = cd1;

Rule 2: If A = a and B = b and C = c
THEN: D = d2 with CF(D) = cd2;

Rule 3: If A = a and (B = b or C = c)
THEN: D = d3 with CF(D) = cd3;

Rule 4: If A = a' or B = b' or C = c'
THEN: D = d4 with CF(D) = cd4;

Table 3: Relations from Production Rules

R1, a relation containing rules 1 and 2

Rule#	IF: A	B	C	D	THEN: CF(D) E CF(E)
1	a	b	c	d1	cd1
2	a	b	c	d2	cd2

R2, a relation containing rules 3 and 4

Rule#	IF: A	B	C	D	THEN: CF(D) E CF(E)
3	a	b		d3	cd3
	a		c	d3	cd3
4	a'			d4	cd4
		b'		d4	cd4
			c'	d4	cd4

The first two rules show that one condition may result in more than one action. Rule 3 shows that different conditions may result in the same action. Production rules with similar IF conditions form a relation.

The key for the rule relations is formed by combining the IF conditions for each tuple. This assures that the resulting relation will be in Fourth Normal Form. Table 3 shows how the production rules from Table 2 may be mapped into the relational model. Notice that in Table 2 the value of D must be included in the primary key because of the identical values of A,B,and C [9].

VII. A self-describing metaschema

Mark and Roussopoulos [10] have proposed an active and integrated data dictionary system which uses the services offered by the DBMS and is flexible enough to control its own evolution. They describe two orthogonal dimensions of data description: the point-of-view dimension and the intension-extension dimension.

The point-of-view dimension consists of three levels of data description: the external, conceptual, and internal schema. These three levels provide data independence.

The intension-extension dimension provides four levels of data description:

1) the application data;

2) the application schema which provides information about specific applications;

3) the data dictionary schema which provides information about the management and use of data;

4) the metaschema which consists of information about the data model.

Each level of description is the intension of the succeeding description and the extension of the preceding one. A description of the metaschema is explicitly stored as part of its own extension.

Using the object-role data model, the authors have mapped the core metaschema into object-oriented tables. The objects in a relational schemata are relations, domains, and attributes. The relation RELN defines relationships between existing relations and their names. ATTN defines relationships between attributes and their names. The relation DOMN describes the domains and the relation RDAS defines the relationships between relations, domains, and attributes. These mappings allow the metaschema to be stored in the database it defines.

The authors further describe a set of operations which control all operations on the data dictionary schema. They state that the operations specified must be explicitly represented in the metaschema itself in order for it to remain self-describing. Because object-oriented data models support storage of procedural information, this task is possible.

VIII. A Proposed Data Dictionary Architecture

In order to satisfy the need for dynamic structuring and closer integration between data and meta-data, it appears that both rules and facts must be actually stored in the database. With this in mind, facts about the data can be stored in relations modelled using the "semantic relation data model" and rules can be stored according to Li's mapping. These two representations match the way in which PROLOG defines predicates.

The core metaschema is designed using the base tables described by Mark and Roussopoulos. These tables have been enhanced by expanding the domain concept as described in section III. Figure 3 is the graph generated by combining these two concepts.

The circles in Figure 3 represent domains. The single boxes represent the actual relations, while the divided boxes represent the attributes associated with each relation.

The resulting relations are shown in Table 4. The relation DOMAIN has been extended to include information necessary for defining attributes in ORACLE, which was the relational product used in developing the prototype. Two additional relations further define the domain: (1) ENU allows enumerated domains for abstract data types such as objects, and (2) FUN stores the name of processes for testing and manipulation of the data. A final core relation RULES stores design rules for inferencing.

Other relations are created or dropped as the associated objects are added to or deleted from the database. When this occurs, the relations RELN, ATTN, and RDAS must be updated, thus ensuring a dynamic metaschema. The following example shows how an object might be added to the database.

In order to add the arch described in Section IV, the user would be asked whether the object was elementary or a composite of other objects. If the object is a composite, its components must be first added and described and discussed in Section III. For example, "BEAM1", "BEAM2", and "BEAM3" would be considered elementary items and must be stored in the database before the arch can be defined.

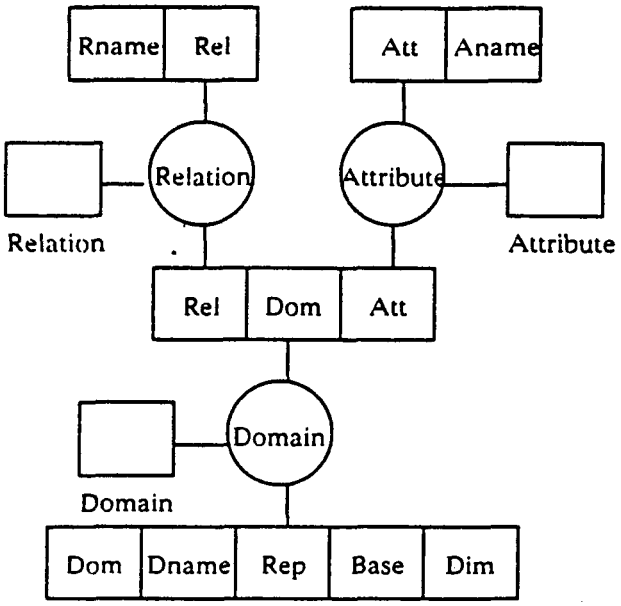

Figure 3: Prototype Core Metaschema

Table 4: Core Relations for Prototype

1) RELN(rname, rel)

RELN	r1
DOMN	r2
:	:

2) DOMN(dom, dname, rep, base, dim)

d1	relation	STA	CHAR(c)	2
d2	attribute	STA	CHAR(c)	2
d3	domain	STA	CHAR(c)	2
:	:	:	:	:

3) RDAS(reln, dom, att)

r1	d4	a1
r1	d1	a2
:	:	:

4) ATTN(att, aname)

a1	rname
a2	rel
:	:

5) ENU(dom, member)

d8	arch
d8	beam1
:	:

6) FUN(dom, process)

d11	TSTAGE
:	:

7) RULES(cond1, cond2, cond3, result)

:	:	:	:

The three beams are particular instances of the type beam which is a member of the domain object. A relation OBJECTS can be created to store these ASA links. The resulting relation is shown in Table 5.

Once the objects are added to the database, the relationships between them can be added. The two relationships SUPPORTS and DISTANCE>0 form two relations as shown in Table 5.

A design rule completes the process of describing the arch:

If SUPPORTS(x,z) and SUPPORTS(y,z) and DISTANCE>0(x,y)
then ARCH(x,y,z);

Each condition evaluates to true if the specified tuple is found to exist. Thus application of this rule to the relations in Table 5 would confirm the fact that the object formed by the three beams is indeed an arch. The arch can be named and added to the OBJECTS relation and the relation CONTAINS can be added to describe the components of the arch. The resulting relations are shown in Table 6.

Table 5: Relations after Adding Beams

1) OBJECTS(oname, otype)

oname	otype
Beam1	beam
Beam2	beam
Beam3	beam

2) SUPPORTS(oname1, oname2)

oname1	oname2
Beam1	Beam2
Beam3	Beam2

3) DISTANCE>0(oname1 oname2)

Beam1	Beam3

Table 6: Relations after Design Rule

1) OBJECTS (oname, otype)

oname	otype
Beam1	Beam
Beam2	Beam
Beam3	Beam
Arch1	Arch

2) CONTAINS (oname1 oname2)

oname1	oname2
Arch1	Beam1
Arch2	Beam2
Arch3	Beam3

IX. Conclusion

CAD/CAM and engineering applications have special needs which are not presently being met by commercial DBMS's. These needs include the ability to represent abstract data types, relationships between the data items, and certain knowledge about the data.

Extension of traditional data dictionaries may be able to meet some of these needs. Extending the concept of domains allows the expression of abstract data types. Storing knowledge rules within the DBMS and using an object-oriented data model allow the representation of complex relationships.

Application of artificial intelligence techniques will allow the data dictionary to become a knowledge manager rather than a data manager. Only then will CAD/CAM and engineering databases be truly effective.

X. References

1. S. Cammarata and M. Melkanoff, "An Interactive Data Dictionary Facility for CAD/CAM Data Bases," Expert Database Systems: Proceedings from the First International Workshop, ed. Larry Kerschberg, Benjamin/Cummings, Reading, Mass., pp.423–440.

2. D.J. Hartzband and F.J. Maryanski, "Enhancing Knowledge Representation in Engineering Databases," Computer. Vol. 18, No. 9, Sept. 1985, pp. 39–48.

3. C.J. Date, An Introduction to Database Systems, Addison-Wesley, Reading, Mass., 1986, p. 39.

4. C. Zaniolo et al., "Object Oriented Database Systems and Knowledge Systems," Expert Database Systems: Proceedings from the First International Workshop, ed. Larry Kerschberg, Benjamin/Cummings, Reading, Mass., 1986, pp. 50–65.

5. J. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, Mass., 1984, p. 304.

6. C. Kellogg, "From Data Management to Knowledge Management," Computer, Vol. 19, No. 1, Jan. 1986, pp. 75–84.

7. M. Brodie et al., "Knowledge Base Management Systems: Discussions from the Working Group," Expert Database Systems: Proceedings from the First International Workshop, ed. Larry Kerschberg, Benjamin/Cummings, Reading, Mass., 1986, pp. 19–33.

8. H. Wedekin, "Supporting the Design of Conceptual Schemata by Database Systems," International Conference on Data Engineering, Apr. 1984, pp. 434-438.

9. A. Shepherd and L. Kerschberg, "Constraint Management in Expert Database Systems," Expert Database Systems: Proceedings from the First International Workshop, ed.Larry Kerschberg, Benjamin/Cummings, Reading, Mass., 1986, pp. 309-331.

10. S. Borkin, Data Models: A Semantic Approach for Database Systems, MIT Press, Cambridge, Mass., 1980, pp. 63-93.

11. R. King, "A Database Management System Based on an Object-Oriented Model," Expert Database Systems: Proceedings from the First International Workshop, ed Larry Kerschberg, Benjamin/Cummings, Reading, Mass., 1986, pp. 443-468.

12. H. Li, "To Develop a Data-knowledge Base Management System by Utilizing Relational Database Management System", in Proceedings of Applications of Artificial Intelligence IV, (A Conference on 15-16 April 1986 in Insbruch, Austria).

13. M. Morgenstern, "The Role of Constraints in Databases, Expert Systems, and Knowledge Representation," Expert Database Systems: Proceedings from the First International Workshop, ed. Larry Kerschberg, Benjamin/Cummings, Reading, Mass., 1986, pp. 351-368.

14. L. Mark and N. Roussopoulos, "Metadata Management," Computer, Vol. 19, No. 12, Dec. 1986, pp. 26-36.