EDICT - An Enhanced Relational Data Dictionary: Architecture and Example

James P. Davis, Senior Knowledge Engineer

NCR Corporation
General Purpose Systems Division
3325 Platt Springs Road
West Columbia, S.C. 29169

Ronald D. Bonnell, Director

Center for Machine Intelligence
University of South Carolina
Columbia, S.C. 29208

ABSTRACT

An integrated data dictionary for a relational DBMS provides a centralized management environment for maintaining information about the data in the database relations. However, current relational data dictionaries do not provide adequate facilities for the capture and representation of high-level semantic information about the enterprise whose data is stored in the tables of the database. This paper describes an approach for enhancing an integrated relational data dictionary in order to capture and faithfully present this high-level enterprise schema in a form which can be incorporated directly into the relational database system. The approach, referred to as *EDICT*, allows for the specification of the enterprise schema for a database as an extension of the data dictionary by storing the schema as a set of normalized tables. In addition, it facilitates more effective use of existing data dictionaries by providing an extended meta-schema which describes the structure and use of the dictionary itself.

1. INTRODUCTION

It has been more than fifteen years since the relational model was first introduced [CODD70], and in that time, many commercial DBMS products built around this model have become powerful tools for defining and managing data of interest to an organization or enterprise. Most, if not all, commercial products provide facilities for data acquisition (via forms), data management (via data dictionary) and data reporting (via reports).

The data dictionary is arguably the most important facility of a DBMS in that it provides the capability of managing and maintaining the consistency and integrity of the data stored in the DBMS, as it is used and modified by a number of users simultaneously. For a given DBMS, a data dictionary generally provides data about both the logical level as well as the physical level.

However, the data dictionaries of most commercial products are somewhat lacking in terms of the scope and the type of data that is made available for use in managing the information stored. First, many of the semantic constraints which are captured in the enterprise domain model are lost in the conversion to the DBMS schema tables. Second, most data provided by the data dictionary is useful for the system to manage the current state of the database, but is not much help in assisting the user in managing the underlying data model, or schema, as its evolves over time. Thus, the user or administrator must handle all schema management outside of the DBMS environment and then incorporate changes manually. Third, for existing databases, the facilities for querying the data dictionary are limited, in that its internal structure, function, and use are opaque to the user; however, there are many instances when a user or application would like to query information about the dictionary catalogs themselves, and not what is in them.

This paper presents an *e*nhanced data *dict*ionary facility, known as *EDICT*, which captures and represents additional knowledge of interest to the database user. EDICT is integrated into the DBMS, utilizes a single representational framework for capturing multiple levels of knowledge about the database, and utilizes a single query language to access its tables--that which is supported by the DBMS (e.g., SQL). In addition, this paper will discuss these extensions in the context of an information architecture which incorporates multiple levels of information into the *e*nhanced *dict*ionary.

The basic premises for this examination of an enhanced data dictionary are:

1. to provide extended representation capabilities to existing data dictionaries in commercially available products;
2. to allow knowledge about the enterprise schema to be captured and represented directly in the database itself, having the ability to be examined and utilized;
3. to allow knowledge about the structure and inner workings of the data dictionary to be captured and represented directly in the database;
4. to provide an environment for developing loosely-coupled knowledge base management systems (KBMS) which provide for the *intelligent* management and control of data and knowledge about data; and,
5. to provide an environment, facilitated by the data dictionary, where the database design activity can be carried out by computer.

The remaining sections of this paper are organized as follows. Section 2 discusses the data dictionary concept as applied in current commercial and research systems relevant to this paper. Section 3 presents the EDICT architecture and its components. Finally, the paper presents an example database application and the corresponding schema tables under EDICT.

The discussion in this paper will use the ANSI/SPARC DAFTG database model as a point of reference for this discussion. In addition, the schemata presented are defined in terms of the E-R data model. It is assumed that the reader is familiar with both.

2. THE DATA DICTIONARY CONCEPT

The data dictionary has been discussed in [ATRE80], [HAWR85], and other sources, as a repository for data, from both logical and physical levels of the database system. This information is useful in describing the contents, organization and use of data stored in the database.

However, one major aspect for the use of the data dictionary is during the schema design process [TSIC82, KAHN85, JARK86]. Here, the data dictionary is used to document functions and data classes, along with relationships and behavior of data in the enterprise. This is useful for purposes of more effective communication among designers, users, and administrators. Most data dictionaries do not directly support this activity.

There has been much work in the area of extending the ideas of what information can and should be captured in the data dictionary, and how this additional information can be used.

The basic premise for most work in this area involves the definition of multiple levels of abstraction, with distinctions between data, meta-data, schema, and meta-schema. Generally, the descriptions at the higher levels defines the structure of the level just below it; i.e., a level is the *intension* and the level below it is the *extension*. The most notable work in this area is the proposed ANSI/SPARC reference model for DBMS architecture [BURN86].

The ANSI/SPARC DBMS reference model, proposed by the Database Architecture Framework Task Group (DAFTG), presents a multilayered architecture for describing the definition and use of data in a DBMS. The description of data is at four levels--*application data*, *application schema*, *data dictionary schema*, and *data model schema*--where each level is the extension (i.e., the "data") of the level above it and is the intension (i.e., the "schema") of the level below it. Furthermore, the levels are self-describing, in that each level's intension is itself stored as part of its extension. This is presented in the context of te ANSI/SPARC 3-schema architecture.

This architecture is referenced and used in [MARK83, MARK87], where the management of metadata is handled by defining operators which facilitate dynamic modification of the data and schemata via changing the contents of the corresponding intension descriptions.

The *EDICT* approach is very similar to the DAFTG architecture; in fact, it conforms to it where appropriate. The major distinction between DAFTG and EDICT is that (as will be shown) the DAFTG model doesn't concern itself with data and schema descriptions which are outside the context of the DBMS environment. The EDICT architecture is primarily concerned with incorporating the enterprise schema into the data dictionary. The enterprise schema is independent of any DBMS implementation and, as such, is an enhancement to the DAFTG architecture.

A similar approach to DAFTG is presented in [GOLD85] for IRDS, which is also a multilevel self-describing data architecture. Other ideas for the inclusion of semantic information into the definition and management functions of a data dictionary are presented in [BRAT83, MART83, RUSP83]. EDICT shares the same underlying representation for schema modeling--the Entity-Relationship Data model [CHEN76].

3. ARCHITECTURE

EDICT is a multilayered architecture for facilitating the development and management of information in a relational DBMS; however, the architecture is generic enough such that it could be applied to other types of DBMS's.

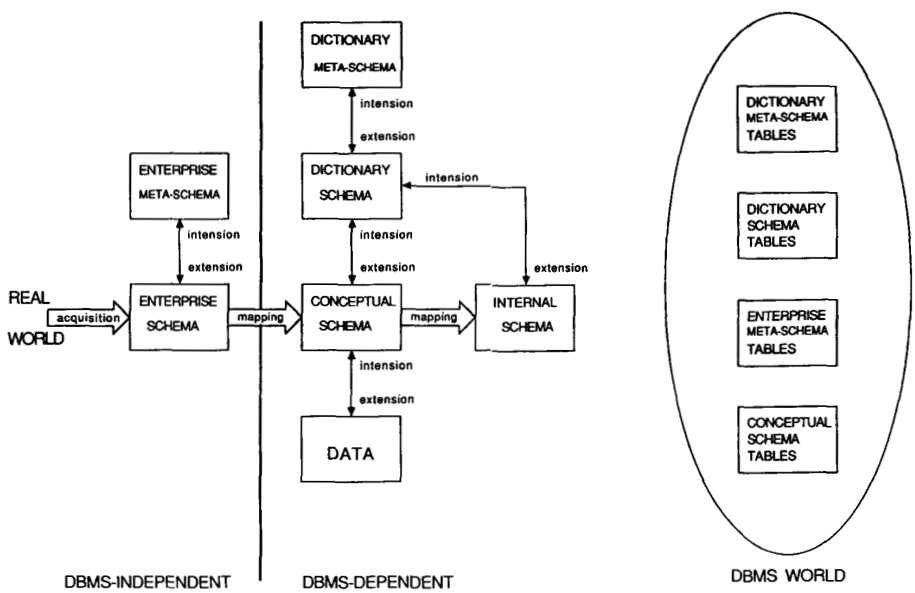

Figure 1. EDICT Architecture and Components

EDICT is defined as enhancements to existing relational data dictionary facilities, through the definition of additional schemata which are integrated into the dictionary proper.

Figure 1 presents the architecture, which has horizontal and vertical dimensions. The horizontal dimension shows the flow of knowledge during the process of designing a database--from the acquisition of enterprise knowledge into an *enterprise schema* (such as can be developed using the E-R model), through the mapping to the *conceptual schema* (which is the relational model), to finally mapping to the *internal schema* (which is the underlying storage structures and indexes).

The vertical dimension shows the levels of abstraction between data and the associated schema which defines its structure, at the level just above it. The *conceptual schema* defines the structure of the enterprise which is capable of being represented in the underlying relational data model. It is derived from a mapping of the *enterprise schema* (e.g., an E-R diagram) according to a methodology such as [DUMP81, BONN84]. The E-R model is not directly represented in the DBMS, but its mapping to the *conceptual schema* is.

The descriptions of the *conceptual schema* are kept in the *dictionary schema* (i.e., data dictionary or system catalogs). The *dictionary schema* contains data about the *conceptual*, *internal*, and *external* schemata (the applicability of the *external schema* is not considered in this paper, but is implicitly incorporated into the EDICT architecture).

Up to this point, the EDICT architecture is compatible with DAFTG, but now EDICT makes a distinction between the *dictionary schema* and the *dictonary meta-schema*. The *dictionary meta-schema* contains data which defines the structure and use of the *dictionary schema*. This goes beyond the notion of having a self-describing schema, but it may be a matter of interpretation as to whether the *dictonary meta-schema* should be a distinct level or not. However, there is considerable value in having this meta-schema available for query by users or applications [RTI85], such that the structure and definition of the data dictionary can be understood.

It should be noted that the DAFTG's *data model schema* level is not covered in this paper, but is implicit in the EDICT architecture. Most commercial data dictionaries incorporate some portion of this schema (i.e., capturing portions of the relational semantics in the relational tables of the data dictionary).

However, a schema which is more useful, under the premises discussed at the outset of this paper, is that of the *enterprise meta-schema*. This schema allows the "essence" of the enterprise schema, captured in a DBMS-independent semantic data model (such as the E-R model), to be represented and preserved in the underlying DBMS data model.

Thus, the EDICT architecture could be considered a subset as well as a superset of the DAFTG reference model--a subset in that it deals explicitly with most of the components of the *intension-extension* dimension, and a superset in that it defines an additional level in the *point-of-view* dimension for the modeling of the enterprise domain.

In the following descriptions of the EDICT components, the E-R data model is used to represent the semantic intent of each of the schemata. With each schema, the E-R representation is mapped to its corresponding representation in the relational model. The schemata presented are by no means complete, but are shown with enough detail to illustrate the concepts.

It should be noted, as a general principle, that the relational model is less expressive than the E-R model and, as such, cannot retain the full semantic intent from the E-R model--thus, information is lost in the mapping. In the context of EDICT, the *enterprise meta-schema* serves the function of preserving some of these semantics which would otherwise be lost.

3.1. Enterprise and Conceptual Schemata

The *enterprise schema* captures the objects and associations which are of interest to the specific enterprise being modeled. In this paper, the enterprise modeling procedure is used to capture information about the enterprise in terms of the constructs of the E-R model--entity sets, relationship sets, attributes, cardinality, etc. The *enterprise schema* is a representation which is independent of any DBMS-specific environment. This is mapped to a specific DBMS environment via the logical data model supported--relational tables. The *enterprise schema* exists as data in the *enterprise meta-schema* tables of the supporting relational DBMS. The schema shown in Figure 2 is for a food-coop example discussed in the next section.

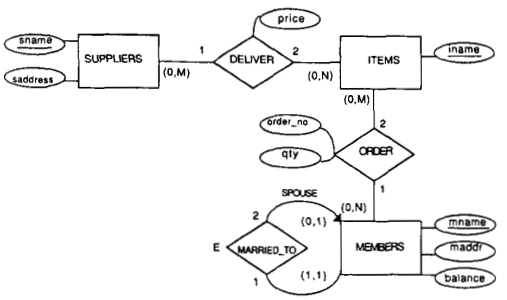

Figure 2. E-R Model of Enterprise Schema

The *enterprise schema* is mapped to a relational schema, which is the *conceptual schema* for a relational database. In a DBMS environment, the relation schemes are created, and the user must enter the data instances. The relational schema for Figure 2 is shown below:

SUPPLIERS(SNAME, ADDR)
ITEMS(INAME)
MEMBERS(MNAME, MADDR, BALANCE, SPOUSE_MNAME)
DELIVER(SNAME, INAME, PRICE)
ORDER(MNAME, INAME, ORDER_NO, QTY)

3.2. Dictionary Schema

The *dictionary schema* is the schema which captures the information required by a given DBMS to manage the *conceptual schema*. This is the defining structure for the particular DBMS's data dictionary, and as such, is dependent on the implementation and organization of the DBMS being used--this is in terms of what "data" about the data is captured and represented by the DBMS.

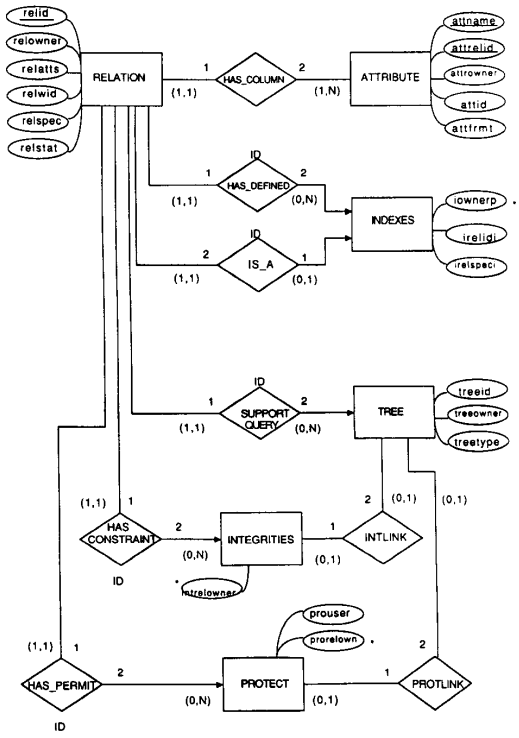

Figure 3. E-R Model of Dictionary Schema

The *dictionary schema* forms the basis of the data dictionary system catalogs. The approach in this paper extends these catalogs by incorporating not only data about the data (hence *meta-data*), where knowledge about the the data in terms of the external, conceptual, and internal schemas of the ANSI reference model is kept, but also incorporates knowledge about the underlying *enterprise schema*, independent of the logical model, stored in tables.

Note that the *dictionary schema* is closely coupled to the particular DBMS being used, whereas the *enterprise meta-schema* and the *enterprise schema* are not.

The tables defined for the meta-data, as supported in a version of the Ingres† DBMS, are shown below (note that this is a partial dictionary schema):

RELATION(RELID, RELOWNER, RELATTS, RELWID, RELSPEC, RELSTAT,...)
ATTRIBUTE(ATTRELID, ATTOWNER, ATTID, ATTNAME, ATTFRMT,...)
INDEXES(IRELIDP, IOWNERP, IRELIDI, IRELSPECI,...)
TREE(TREERELID, TREEOWNER, TREEID, TREETYPE,...)
PROTECT(PRORELID, PRORELOWN, PROUSER, PROTREE,...)
INTEGRITIES(INTRELID, INTRELOWNER, INTTREE,...)

3.3. Enterprise Meta-Schema

The *enterprise meta-schema* captures the objects and associations which are directly modeled using the E-R data model--explicitly as a self-describing E-R model (i.e., an E-R model of the E-R model itself). This E-R representation captures and expresses the concepts and structure, as well as the objects and associations, inherent in the E-R semantic data model and its use.

Figure 4 depicts this representation of a partial *enterprise meta-schema*. Additional information which would be kept in this schema would be information about synonyms, keys, etc. Several new notational constructs are used, as defined in [DAVI87]. The "))XOR" construct denotes that an attribute is either associated with an entity set or a relationship set, but not both and not neither. The integer numbers beside the diamond constructs indicate the ordering of entity sets in the context of the relationship that they participate in.

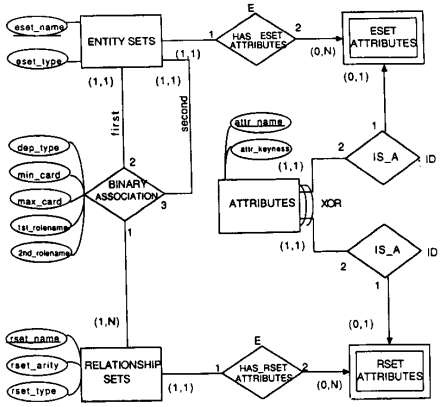

Figure 4. E-R Model of Partial Enterprise Meta-Schema

This schema captures the information in the specification and use of the E-R model and, as such, is a schema which is independent of any connection to a particular DBMS environment or logical data model. In other words, this schema captures information which is not connected with how data or meta-data will be organized, either logically or physically, in any given DBMS.

Even though this schema captures concepts which are independent of any DBMS, it must be capable of being represented in the underlying DBMS; thus, the *enterprise meta-schema* is mapped to a set of relational tables according to a well-understood algorithm [DUMP81, BONN84]. These tables are defined below.

ENTITY_SETS(ESET_NAME, ESET_TYPE)
RELATIONSHIP_SETS(RSET_NAME, RSET_ARITY, RSET_TYPE)
ATTRIBUTES(ATTR_NAME, ATTR_DOMAIN, ATTR_KEYNESS)
ESET_ATTR(ATTR_NAME, ESET_NAME)
RSET_ATTR(ATTR_NAME, RSET_NAME)
BINARY_RELATIONSHIP(RSET_NAME, FIRST_ESET_NAME,
SECOND_ESET_NAME, DEP_TYPE,
MIN_CARD, MAX_CARD,
FIRST_ROLENAME, SECOND_ROLENAME)

It should be noted that the choice of the E-R model as the basis of this meta-schema is due to the preference of the authors; other semantic data modeling formalisms could be used instead. Also, the other part of enterprise analysis, transaction analysis [TSIC82], where queries on the database are examined, is not considered in the scope of this paper.

3.4. Dictionary Meta-Schema

The *dictionary meta-schema* is the schema used to capture and represent the meta-data about the structure and organization of the *dictionary schema* (i.e., the data dictionary or system catalogs). This schema presents knowledge about the organization and definition of the components of the data dictionary of a specific DBMS once mapped to tables. The E-R representation for this schema is shown in Figure 5.

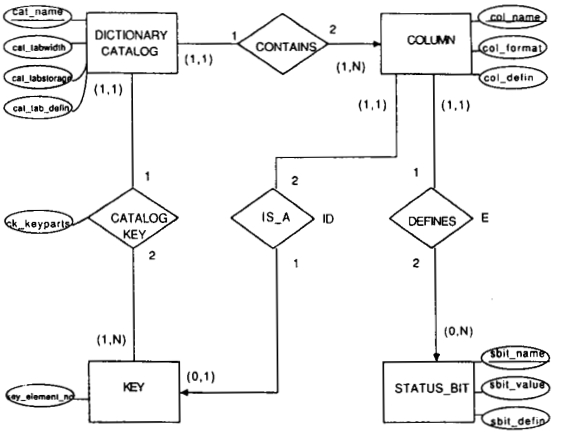

Figure 5. E-R Model of Dictionary Meta-Schema

The corresponding relational tables for the *dictionary meta-schema* are given below:

CATALOG(CAT_NAME, CAT_TABWID, CAT_TABSTORAGE,
CAT_TAB_DEFIN)
COLUMN(COL_NAME, COL_FORMAT, COL_DEFIN, CAT_NAME)
STATUS_BIT(SBIT_NAME, SBIT_VALUE, SBIT_DEFIN, COL_NAME)
CATALOG_KEY(COL_NAME, KEY_ELEMENT_NO, CAT_NAME,
CK_KEYPARTS)

4. EXAMPLE

An example can be used to best illustrate how the EDICT approach can be used to enhance the data dictionary. A food-coop database, after [ULLM82] is to be set up, which keeps track of the members of the coop, the suppliers, items ordered, etc. The E-R model of the *enterprise schema* was shown in Figure 2; the corresponding relational tables, instantiated with data, are shown below.

SUPPLIERS

sname	saddress
Sunshine Produce	16 River St.
Purity Foodstuffs	180 Industrial Rd.
Tasti Supply Co.	17 River Dr.

ITEMS

iname
granola
unbleached flour
whey
sunflower seeds
lettuce

MEMBERS

mname	maddr	balance	spouse_mname
Brooks, B.	7 Apple Rd.	10.50	Hart, W.
Field, W.	43 Cherry Ln.	0	Field, S.
Robin, R.	12 Heather Rd.	-123.45	-
Hart, W.	7 Apple Rd.	-43.00	Brooks, B.
Field, S.	43 Cherry Ln.	-25.00	Field, W.

DELIVER

sname	iname	price
Sunshine Produce	granola	1.29
Sunshine Produce	lettuce	.89
Sunshine Produce	sunflower seeds	1.09
Purity Foodstuffs	whey	.70
Purity Foodstuffs	granola	1.25
Tasti Supply Co.	lettuce	.79

ORDER

mname	iname	order_no	qty
Brooks, B.	granola	1042	5 boxes
Brooks, B.	unbleached flour	1042	10 lbs.
Robin, R.	granola	1045	3 boxes
Hart, W.	whey	1044	5 lbs.
Robin, R.	sunflower seeds	1039	2 lbs.
Robin, R.	lettuce	1039	8 heads

The *dictionary schema* is instantiated with the meta-data concerning this database. This schema is demonstrated based on information published about the structure of the system catalogs for a version of the Ingres† DBMS [DUER83]. Two of these tables are shown below for the dictionary schema of Figure 3:.

RELATION

relid	relowner	relatts	relwid	relspec	relstat	...
suppliers	jd	2	40	5	0600134	...
items	jd	1	20	5	0600134	...
members	jd	3	44	5	0600134	...
deliver	jd	3	44	5	0600134	...
orders	jd	4	48	5	0600134	...

ATTRIBUTE

attrelid	attrowner	attid	attname	attfrmt	...
suppliers	jd	1	sname	32 (character)	...
suppliers	jd	2	saddress	32 (character)	...
items	jd	1	iname	32 (character)	...
members	jd	1	mname	32 (character)	...
members	jd	2	maddr	32 (character)	...
members	jd	3	balance	5 (money)	...
deliver	jd	1	sname	32 (character)	...
deliver	jd	2	iname	32 (character)	...
deliver	jd	3	price	5 (money)	...
orders	jd	1	name	32 (character)	...
orders	jd	2	iname	32 (character)	...
orders	jd	3	order_no	30 (integer)	...
orders	jd	4	qty	30 (integer)	...

The important additions for enhancing the data dictionary for this database come with instantiation of the enterprise and dictionary meta-schemata. The *enterprise meta-schema* provides information about the initial enterprise model, and the semantic constraints associated with it, which can not be captured in the relational schema. The *enterprise meta-schema* allows this information to be preserved.

RELATIONSHIP_SETS

rset_name	rset_arity	rset_type
deliver	binary	regular relationship set
orders	binary	regular relationship set
married_to	unary	regular relationship set

ATTRIBUTES

attr_name	attr_domain	attr_keyness
name	*alphanumeric*	key
address	alphanumeric	non-key
balance	real number	non-key
item	alphanumberic	key
sname	alphanumeric	key
saddress	alphanumeric	non-key
order_no	integer	non-key
quantity	integer	non-key
price	real number	non-key

BINARY_RELATIONSHIP

rset_name	1st_ename	2nd_ename	dep_type	min_card	max_card	1st_rlname	2nd_rlname
deliver	suppliers	items	-	0:1	M:N	-	-
orders	members	items	-	0:1	M:N	-	-
married_to	members	members	existence	1:0	1:1	-	spouse

ENTITY_SETS

eset_name	eset_type
suppliers	regular entity set
items	regular entity set
members	regular entity set

Populating the ESET_ATTR and RSET_ATTR tables with data from the example is intuitive, and thus is omitted.

An instantiated *dictionary meta-schema* is shown below for the food-coop example. This schema is significant in that it allows users or applications to query for information about the structure of the data dictionary itself.

CATALOG_KEY

col_name	key_element_no	cat_name	ck_keyparts
relid	1	relation	1 part
attrelid	1	attribute	2 parts
attrowner	2	attribute	2 parts
.	.	.	.
.	.	.	.
.	.	.	.

STATUS_BIT

sbit_name	sbit_value	sbit_defin	col_name
s_catalog	0000001	table is a system catalog	relstat
s_noupdt	0000002	no updates allowed on table	relstat
s_protups	0000004	"protect" catalog reference	relstat
s_integ	0000010	"integ" catalog reference	relstat
s_concur	0000020	*concurrency control via page lock*	relstat
s_view	0000040	user table is a view	relstat
.	.	.	.
.	.	.	.
.	.	.	.

COLUMN

col_name	col_format	col_defin	cat_name
relid	c12	database table name	relation
relowner	*c2*	table owner (2 letter code)	relation
relatts	i2	number of columns in table	relation
relwid	i2	tuple row width (bytes)	relation
relstat	i4	status bits for table	relation
attrelid	*c12*	*table name to which column belongs*	attribute
attrowner	*c2*	owner of table	attribute
attid	i2	column number in table	attribute
.	.	.	.
.	.	.	.
.	.	.	.

CATALOG

cat_name	cat_tabwid	cat_tabstorage	cat_tab_defin
relation	52	hash	describes how each table is stored in database
attribute	35	hash	describes information about columns in tables
indexes	33	hash	contains info. on secondary indexes
tree	120	hash	contains query trees needed for query optimization
integrities	33	hash	contains info. on table integrity constraints

5. SUMMARY

This paper has presented EDICT--an architecture for an enhanced data dictionary facility for a relational DBMS. An example using Ingres† was also presented for a small application.

The EDICT architecture was shown to be highly compatible with the ANSI/SPARC DAFTG database reference model, with some additions. The main distinctions of the EDICT approach are:

1. The *enterprise schema*, captured via the use of a semantic data model, can be stored in the DBMS through the use of an *enterprise meta-schema*. This facilitates having greater insight into the semantics of the enterprise which aren't available in the relational representation alone. Furthermore, it would be possible to perform schema evolution at the semantic level, which is closer to the representation that humans use, rather than in a logical data model that lacks semantic notational constructs (this notion of schema evolution is outside the scope of this paper, but has been considered by others, such as [MARK87]).
2. The existing system catalogs (i.e., the *dictionary schema*) can be more effectively used by having descriptions of their structure and meaning available to users and applications. This is facilitated by the *dictionary meta-schema*.
3. In the context of the EDICT architecture, the enhanced dictionary is actually composed of the *dictionary schema*, the *dictionary meta-schema*, and the *enterprise meta-schema*. All are represented in the same underlying data model supported by the DBMS, and all can be queried using the same query language supported by the DBMS.

The EDICT architecture was demonstrated for a relational DBMS, but is also valid for hierarchical or network-based DBMS's as well. In addition, the E-R model was used for defining the *enterprise meta-schema*, but any high-level semantic data model could be used.

It should be noted that, in the example presented, EDICT was not shown to be self-describing, after the fashion of DAFTG [BURN86, MARK87] (i.e., schema tables which contain data about themselves). But the schemata could be self-describing, where it makes sense to do so, provided that the supporting DBMS allows self-describing of a schema (such as the *dictionary schema*in this paper). Most commercial DBMS products don't allow this, but it appears that Oracle‡ does provide the ability for a user to extend the data dictionary [ORAC86]. This is a part of further investigation.

The EDICT architecture is the basis of research in the areas of loosely-coupled knowledge base management systems, and in the development of model-based reasoning systems to perform logical database design [DAVI85]. Endowing EDICT with the capability to reason about the state of its schemata is the subject of further research.

6. BIBLIOGRAPHY

[ATRE80]
Atre, S., *Data Base: Structured Techniques for Design, Performance, and Management*, John Wiley and Sons, Inc., 1980.

[BONN84]
Bonnell, R.D., *Tower-1632 Ingres Relational DBMS Course Notes*, NCR Corporation, 1984.

[BURN86]
Burns, T., E. Fong, D. Jefferson, R. Knox, L. Mark, C. Reedy, L. Reich, N. Roussopoulos, and W. Truszkowski, "Reference Model for DBMS Standardization",
SIGMOD Record, Vol. 15, No. 1, March 1986.

[BRAT83]
Brathwaite, K. S., "An Implementation of a Data Dictionary to Support Databases Designed Using the Entity-Relationship (E-R) Approach", in Davis, C.G., et al (eds.), *Entity-Relationship Approach to Software Engineering*, ER Institute, 1983, pp. 393-410.

[CHEN76]
Chen, P.P., "The Entity-Relationship Model: Toward a Unified View of Data", *ACM Trans. Database Systems 1*, pp. 9-36.

[CODD70]
Codd, E.F., "A Relational Model For Large Shared Data Banks", *Comm. ACM 13*, 1970, pp. 377-387.

[DAVI85]
Davis, J.P., and R.D. Bonnell, "DBA - An Expert DataBase Assistant: Architecture and Specification", Center for Machine Intelligence, University of South Carolina, *USCMI Report #85-30*, 1985.

[DAVI87]
Davis, J.P., and R.D. Bonnell, "Role Representation in the E-R Model", Center for Machine Intelligence, University of South Carolina, *USCMI Report #87-40*, 1987.

[DUER83]
Duerr, S., "INGRES System Catalogs", from *RTI Application Notes #9*, © 1983, Relational Technology Inc.

†Ingres is a trademark of Relational Technology Inc.

‡ Oracle is a trademark of Oracle Corporation

[DUMP81]
Dumpala, S.R., and S.K. Arora, "Schema Translation Using the Entity-Relationship Approach", in Chen, P.P (ed.) *Entity-Relationship Approach to Information Modeling and Analysis*, ER Institute, 1981, pp. 583-608.

[GOLD85]
Goldfine, A., "The Information Resource Dictionary System", *Proceedings of 4th Conference on the E-R Approach*, IEEE Press, 1985, pp. 114-122.

[HAWR84]
Hawryszkiewycz, I. T., *Database Analysis and Design*, Science Research Associates, Inc., Chicago, Ill., 1984.

[JARK86]
Jarke, M., "Search and Knowledge Acquisition in Large KBMS", in Brodie, M. and J. Mylopoulos (eds.), *On Knowledge Base Management Systems*, Springer-Verlag, 1986, pp. 508-522.

[KAHN85]
Kahn, B.K., "Requirement Specification Techniques", in Yao, S.B. (ed.) *Principles of Database Design - Volume 1 Logical Organizations*, Prentice-Hall, Inc., 1985, pp. 1-65.

[MARK83]
Mark, L., and N. Roussopoulos, "Integration of Data, Schema, and Meta-Schema in the Context of Self-Documenting Data Models", in Davis, C.G., et al (eds.) *Entity-Relationship Approach to Software Engineering*, ER Institute, 1983, pp. 585-602.

[MARK86]
Mark, L., and N. Roussopoulos, "Metadata Management", *IEEE Computer*, December 1986, pp. 26-36.

[MART83]
Marti, R.W., "Integrating Database and Program Descriptions Using and E-R Data Dictionary", in Davis, C.G., et al (eds.), *Entity-Relationship Approach to Software Engineering*, ER Institute, 1983, pp. 377-392.

[ORAC86]
The Oracle Database Administrator's Guide (Version 5.0), Oracle Corporation, Part No. 3601-V5.0.

[RTI85]
User Survey Analysis, © 1985, Relational Technology Inc.

[RUSP83]
Ruspini, E.H., and R. Fraley, "ID: An Intelligent Information Dictionary System", in Davis, C.G., et al (eds.), *Entity-Relationship Approach to Software Engineering*, ER Institute, 1983, pp. 367-376.

[TSIC82]
Tsichritzis, D.C., and F.H. Lochovsky, *Data Models*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1982.

[ULLM82]
Ullman, J.D., *Principles of Database Systems*, 2nd. ed., Computer Science Press, 1982.