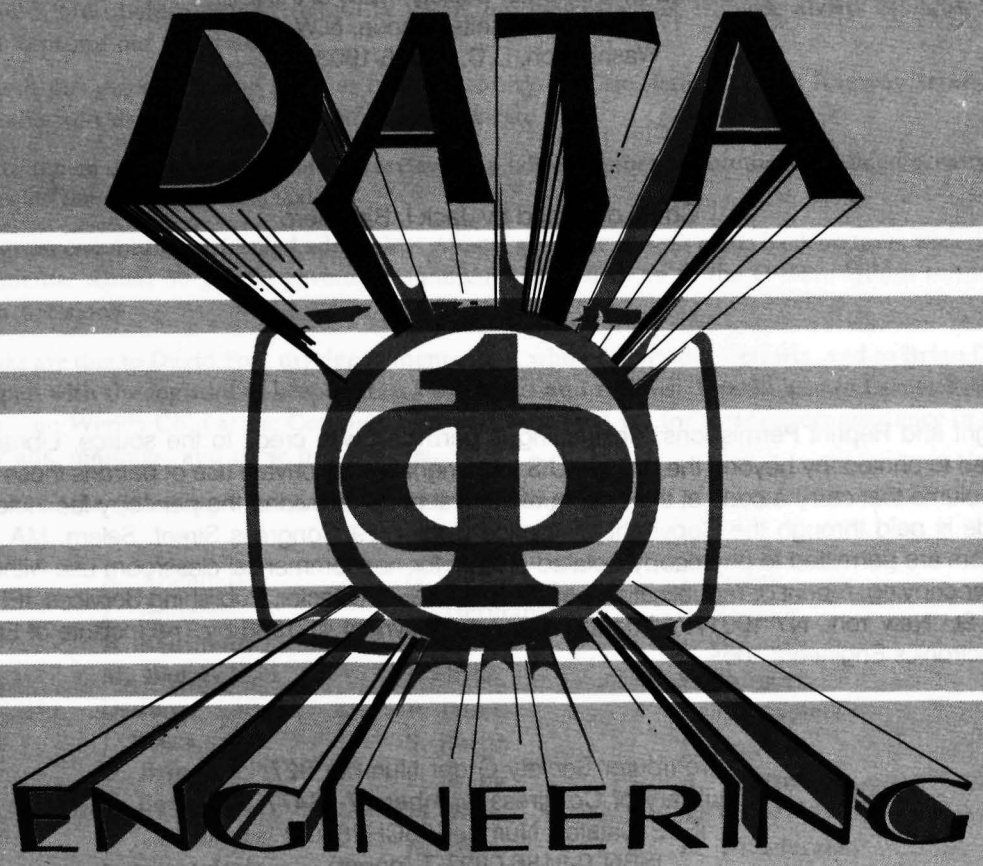
Proceedings

Fourth International Conference on DATA ENGINEERING



February 1-5, 1988
Los Angeles Airport Hilton and Towers
Los Angeles, California, USA

Computer Society Order Number 827
Library of Congress Number 87-83477
IEEE Catalog Number 88CH2550-2
ISBN 0-8186-0827-7
SAN 264-620X

THE COMPUTER SOCIETY OF THE IEEE
IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
COMPUTER SOCIETY PRESS

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and are published as presented and without change, in the interests of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors, Computer Society Press of the IEEE, or The Institute of Electrical and Electronics Engineers, Inc.

Published by Computer Society Press of the IEEE
1730 Massachusetts Avenue, N.W.
Washington, D.C. 20036-1903

Cover designed by Jack I. Ballestero

Computer Society Order Number 827
Library of Congress Number 87-83477
IEEE Catalog Number 88CH2550-2
ISBN 0-8186-0827-7 (paper)
ISBN 0-8186-4827-9 (microfiche)
ISBN 0-8186-8827-0 (case)
SAN 264-620X

Order from: Computer Society of the IEEE
Terminal Annex
Post Office Box 4699
Los Angeles, CA 90080

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331

Computer Society of the IEEE
Avenue de la Tanche, 2
B-1160 Brussels
BELGIUM

IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

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.

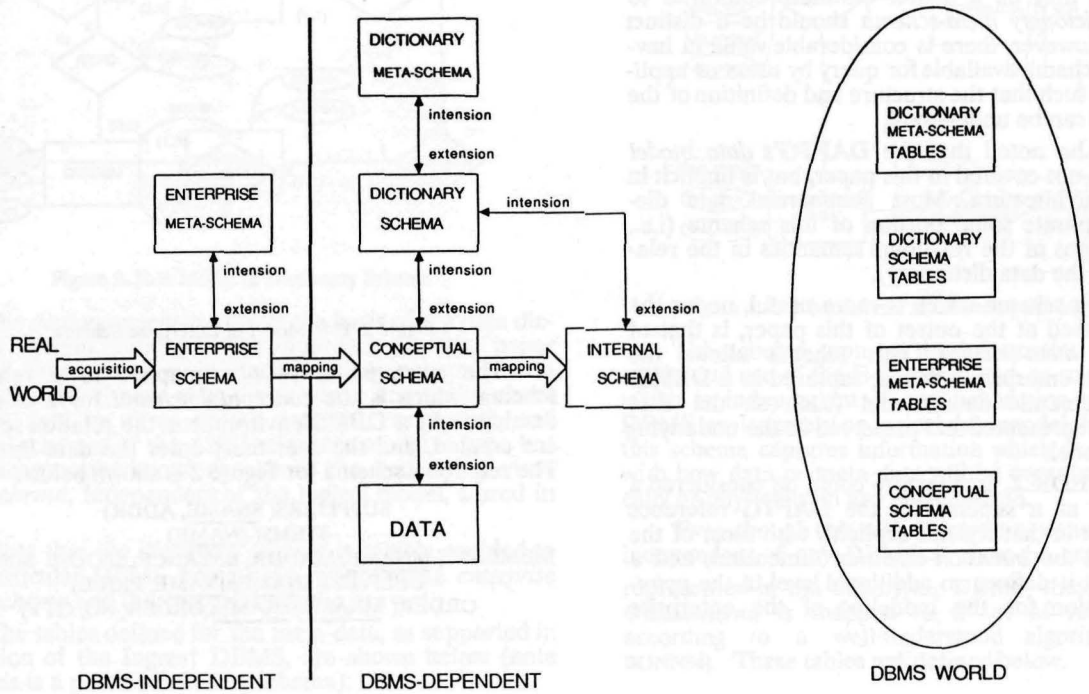


Figure 1. EDICT Architecture and Components

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.

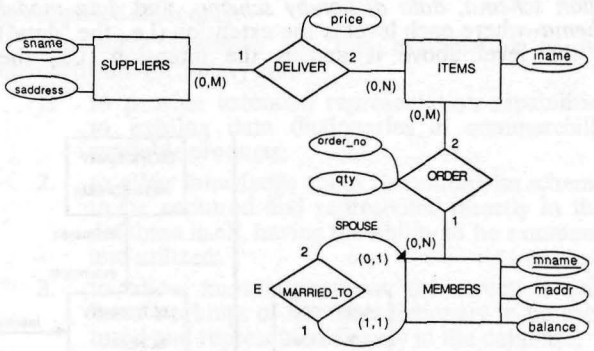


Figure 2. E-R Model of Enterprise Schema

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)

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.