Research

Role of Data Dictionaries in Information Resource Management *

Shamkant B. Navathe
Database Systems Research and Development Center, Computer & Information Sciences Dept., University of Florida, 512 Weil Hall, Gainesville, Fl 32611, USA

and

Larry Kerschberg
College of Business Administration, Institute of Information Management, Technology and Policy, University of South Carolina, Columbia, SC 29208, USA

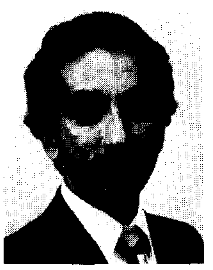
The role of information resource dictionary systems (data dictionary systems) is important in two important phases of information resource management:

First, information requirements analysis and specification, which is a complex activity requiring data dictionary support: the end result is the specification of an "Enterprise Model," which embodies the major activities, processes, information flows, organizational constraints, and concepts. This role is examined in detail after analyzing the existing approaches to requirements analysis and specification.

Second, information modeling which uses the information in the Enterprise Model to construct a formal implementation independent database specification: several information models and support tools that may aid in transforming the initial requirements into the final logical database design are examined.

The metadata – knowledge about both data and processes – contained in the data dictionary can be used to provide views of data for the specialized tools that make up the database design workbench. The role of data dictionary systems in the integration of tools is discussed.

Keywords: Database, Database Design, Logical Database Design, Data Dictionaries, Information Modeling, Enterprise Modeling, Semantic Data Models, Requirement Analysis, Requirements Specification, Information Resource Management, Metadata Management.



Shamkant Navathe is an associate professor with the Database Systems Research and Development Center and the Department of Computer and Information Sciences at the University of Florida, Gainesville. He holds a Ph.D. in industrial and Operations Engineering from the University of Michigan, Ann Arbor. Prior to his Ph.D., he worked for IBM World Trade Corporation and Electronic Data Systems as a System Engineer. During 1975–1979, he was on the faculty of New York University's Graduate School of Business Administration.

He is internationally known for his research work in the areas of database modeling conversion, logical design and distributed database design. He has been active in the areas of CAD/CAM, Expert Systems, and has worked with the OBE project in Office Systems at IBM Research. He has also published papers on statistical database modeling, database mapping and integration, data dictionaries, and query optimization. He is currently writing a textbook, *Fundamentals of Database Systems* (Benjamin Cummings, 1986) with Prof. Elmasri of the University of Houston.

He has done consulting work with Siemens, Nixdorf, and GMD in West Germany and with CCA, Honeywell and ADR, etc., in the U.S. He was the Program Chairman of the 1985 ACM-SIGMOD International Conference on Management of Data at Austin, Texas, and is an Associate Editor of the Journal *Data and Knowledge Engineering* (North-Holland, Amsterdam and New York).

Larry Kerschberg is Associate Professor of Management Science at the University of South Carolina. He also holds the position of Adjunct Associate Professor of Computer Science. Dr. Kerschberg serves as a Faculty Associate to the Institute of Information Management, Technology and Policy, and is Editor of the Institute's Technical Report Series. Prior to his joining the University of South Carolina, Professor Kerschberg was a Member of Technical Staff, Research Division, Bell Laboratories, Holmdel, New Jersey. His previous teaching

* This material represents a revised version of the edited deliberations of the 'Logical Database Design' panel (organised and chaired by Navathe) at the Database Directions III workshop, October 1980. (Sponsors: The National Bureau of Standards and the Association for Computing Machinery.) The original report was published in the NBS Special Publication 500-92. The material originated from the discussions and reports of the Information Requirements and Information Modeling Groups. Participants were M. Brodie, P. Buneman, R. Curtice, J. Fry, B. Housel, D. Jefferson, B. Kahn, L. Kerschberg, D. McLeod, J. Weldon and G. Wiederhold. The authors wish to record and acknowledge their contribution to this article.

positions were with the University of Maryland and the Catholic University of Rio de Janeiro, Brazil. Dr. Kerschberg received his Ph.D. in Engineering from Case Western Reserve University.

Dr. Kerschberg's major research and teaching interests are in the management of information systems, information resource management, enterprise and data modeling, query processing, database management, computer security, and knowledge-based systems. He has published in the *ACM Transactions on Database Systems*, and in leading conferences. Dr. Kerschberg recently served as Program Chairman of the First International Workshop on Expert Database Systems, and is editing a book, *Expert Database Systems*. He is also the Program Chairman for the First International Conference on Expert Database Systems, April 1-4, 1986.

1. Introduction

This paper addresses the issues of information resource management as they relate to an understanding of the user's information requirements, techniques for representing and integrating the individual information models, and the overlap between logical and physical design.

The conclusions are:

i) Logical database design is a complex activity. In large organizations, computer-assisted methodologies will allow designers to cope with this problem more effectively. Use of semantic data models must be stressed.

ii) Requirement analysis is an extremely important phase of logical database design. Existing tools and methodologies have desirable features, but no single technique stands out.

iii) Commerical software for schema design as well as prototype research tools are either inadequate or unusable in realistic situations. The "tool workbench" concept is recommended as a means of integrating these tools.

iv) Data dictionary systems (DDSs) will aid in better database design.

A number of recommendations are also made for distributed databases and allied problems of communication, control, etc.

To set a framework for discussion, the overall logical database design process is shown in Fig. 1, where the various models of data that represent the real world while transforming it to a stored operational database are shown.

Fig. 2 shows the five phases of requirements analysis and planning. Single arrows show sequences of action, while double arrows show mapping among models.

The *Enterprise Model* comprises data expressed in terms of objects (things, activities, events, policies, concepts, etc.) for the overall enterprise plus the attributes and relationships among these objects. As an example, consider a university having a universe of discourse of instructors, students, physical facilities (including building and

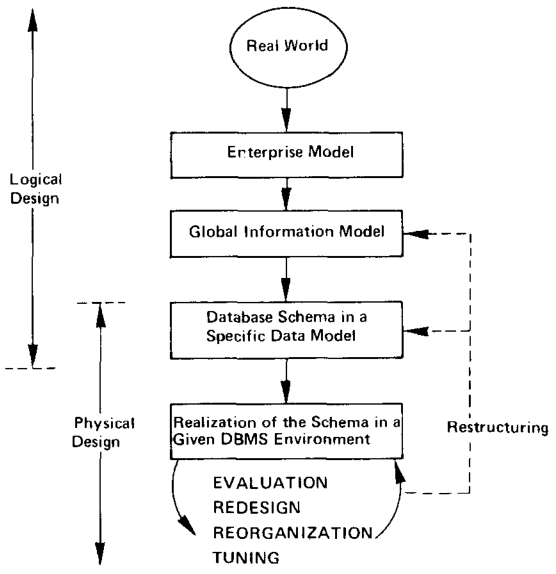


Fig. 1. A Database Design Framework.

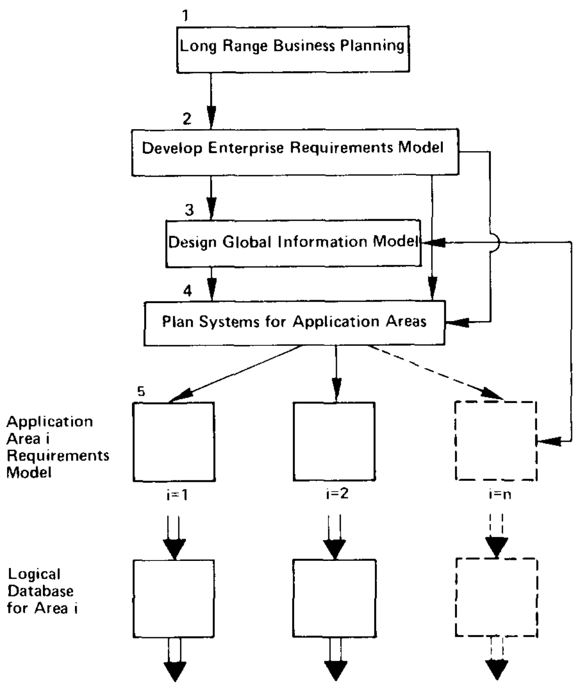

1
Long Range Business Planning
2
Develop Enterprise Requirements Model
3
Design Global Information Model
4
Plan Systems for Application Areas
5
Application Area i Requirements Model
i=1
i=2
i=n
Logical Database for Area i

Fig. 2. Requirements Analysis and Planning in an Information Resource Management Environment.

equipment), courses, etc. The enterprise schema would include all the above objects and associations such as:

- a student enrolls in a course;
- a course is taught by an instructor in a certain room;
- each piece of equipment has a fixed maintenance schedule;
- instructors teach labs (special classes) – meeting in rooms designated as a lab; etc.

These are a few of the possible relationships that may need to be stored in the database. The Enterprise Model sometimes includes the required type of processing. The Enterprise Model should contain relatively static or invariant data.

The *Application Requirement Models* capture the dynamics of the enterprise by developing requirement specifications of individual applications. In the case of the university, this may refer to applications such as financial planning, registration, generation of class rosters, faculty office assignments, classroom assignments, preparation of inventories of equipment by buildings, departments, or rooms, etc., using successive refinements in a top-down design methodology of specification, the applications are described with an increasing level of detail.

The *Global Information Model* results from the integration process; this integrates the Enterprise Model with the individual application models. It should typically use a semantic or database abstraction model and is rather utopian in its ideal form. Organizations where user groups place widely varying demands on the data and want to see their "own (not necessarily compatible) versions" may have to settle for more than one global information model.

For organizations distributing the control of their data to local user groups, the view integration activity proceeds first at the local level and then a global information model may or may not be constructed. Whether a global information model is needed for distributed data management depends upon the amount of "coupling" among local databases, i.e., the sharing of data.

The next phase (*schema design* (SD)) starts from the global information model and ends with the schema for a specific DBMS environment: definition of the DDL, the integrity constraints, the privacy and security constraints, and the processing of the database in that DBMS. This phase overlaps the following:

a) the process of view integration, which is responsible for generating the global information model.

b) the physical mapping of a DBMS schema into the corresponding files and/or storage structures in the DBMS.

It is difficult to draw well-defined boundaries between physical and logical database design. In the later phases of logical database design, to compare alternative designs, it sometimes becomes necessary to capture the targeted physical environment in terms of some broad parameters (e.g., as in [20]).

2. Requirement Analysis for Information Resource Management

2.1. Introduction

Database design, like any other design activity, is critically dependent on a good statement of the enterprise requirements. We are concerned with requirements that affect the logical database design effort; moreover, we address this issue within the context of an Information Resource Management (IRM) philosophy:

- Information is to be regarded as a resource to the organization.
- The information resources should be shared across application and organizational divisions.
- Information consistency, privacy and security are of major importance.
- Data Management is to be independent of specific computerized application systems.
- Computerized application systems must be integrated. A major function within a business unit will constitute one application system.

Two very critical implications follow from this:

a) Under IRM the requirements definition activities *must* be undertaken within the context of a top-down business planning process that relates information to business objectives. Without this, there are no objectives. Currently, information is related to tactical business objectives, although it should be directed at strategic ones.

b) Under IRM the following aspects of the requirements phase must be centralized:

- Coordination of requirements activities,
- Access to requirements documentation,

- Specification of the requirements definition methodology,
- Review and approval of requirements results, and
- Adjudication of disputes.

It is possible to apply IRM to less than an entire organization, but there is some minimum group; e.g., sharing data between two computer programs is not an example of IRM.

2.2. The Activities of the Requirements Phase

A. Construct Long-Range Business Plan

This provides critical inputs that define the nature of the organizational mission, the long-range objectives of the organization, and the data for these. Additionally, it identifies assumptions and constraints important to the business; e.g., an assumption that the Federal Trade Commission will not relax its consumer protection regulations concerning mail order businesses.

B. Develop Enterprise Requirements Model

This first task in the requirements phase provides a high level description of the objects, relationships, and functions within the enterprise. This description serves as a top-down global constraint based on the business plan. Since it is concerned with aspects fundamental to the organization, it should be invariant for a long time. However, it may change as the result of integrating two functional areas (e.g., the result of a merger).

The Enterprise Requirements Model provides:

a) The set of named object types of fundamental interest
b) A definition of each object type and subtypes (with the membership criteria)
c) Relationships (named when appropriate) of fundamental interset between object types.
d) Definition of high-level business functions and their object types, etc.

For example, Holiday Hotels might require PERSONS, with subtypes CUSTOMERS and EMPLOYEES, and ROOMS and CUSTOMER-BILLING-ACCOUNTS as fundamental objects of interest. CUSTOMERS may be involved in several relationships with ROOMS including RESERVATION and OCCUPANCY. CUSTOMERS are also related to the respective BILLING-ACCOUNTS. The fundamental business function of RENTING involves these objects and relationships.

The Enterprise model should ignore details of temporary or local interest only. It has three activities: data collection, specification, and analysis.

C. Design (Initial) Global Information Model

The Global Information Model differs from the Enterprise Requirements model in that the former represents objects and relationships as computerized data. Object identification schemes, the nature of the mapping between objects and their identifiers, domain integrity constraints, etc., are in the purview of the Global Information Model.

D. Plan Application Area Systems

This activity determines the application systems to be implemented within the various application areas. It sets the priority and sequence of these efforts and sets off the requirements effort of each area.

E. Develop Requirements Models for Application Areas

This consists of the same three activities (data collection, specification, and analysis) as the Enterprise Requirements effort, but the objectives are somewhat different. The purpose of the Application Area model is to provide:

- the set of all named object types of interest in an application area.
- definitions for these object types, plus their computerized representation, plus an approximate number of instances of each object type.
- all relationships among those object types in the application area, plus the nature of the mapping (one-to-many vs. many-to-may, optional vs. mandatory, etc.).
- information processing requirements of the application area, including output data levels of summarization, data and processing interactions, processing frequencies, cycles, etc.

An application area is broad and may involve a complex set of interested systems, to be implemented in a phased approach over time. The model provides data for the logical database design activity.

The requirements formulation should be both "bottom-up" (from examination of detailed user needs) and "top-down" (from constraints imposed by the Enterprise Requirements and Global Information Models). The mixture is critical. The development of a complete top-down design has proven to be too time consuming and difficult.

Management will typically not invest in such a large undertaking with a limited immediate payoff. But IRM implies that an enterprise level view be developed and used to ensure the sharing and consistency of data; this cannot result from "merging" or "integrating" independently derived lower level views. The challenge is to include only the necessary levels of detail in the Enterprise and Global level models.

2.3. *State of the Art*

The requirements phase is composed of three activities – requirements collection, requirements specification and requirements analysis. These activities are conducted and accomplished by various individuals, including users, analysts and designers. The characteristics of the individuals are a determining factor in the approach of the requirements phase and the techniques employed.

There are four major approaches: top-down, bottom-up, backward-forward (output driven), and activity analysis (process driven).

A *top-down approach* views the organization as a whole and decomposes it. This can be used to collect information and processing requirements either independently or simultaneously.

The *bottom-up approach* is based on the assumption that modules or programs are the basic elements of any information system. It is assumed to grow in response to the need to add new processing requirements or programs. Bottom-up analysis is based solely on the definition of known processing requirements.

The *backward-forward approach* (output driven) is based on the input-process-output view of a system. This is called backward-to-forward since it begins with the identification of outputs. For each process, the data utilized (inputs and internal data) are identified; this is usually in a top-down manner. First, the high-level data stream is determined; then the output is decomposed into its constituent data items and a backward flow is determined. Subsequently, data items are related to processes and to the sources of data.

Activity analysis (process driven) is also based on the input-process-output model. The analysis begins with the identification of both manual and automated processes (system activities). For each process, the input(s) and the source of each are determined. Additionally, the output(s) produced and their destinations are identified.

Table 1
Data Collection Techniques

Technique	Input	Comments
Review Existing Documentation	System documentation: Management overviews, System output, Functional Specification. Organizational documentation: Organization Charts, Job Descriptions, Policies.	Good as first step in collection. Can identify other appropriate techniques and their scope.
Observe Operating Environment	Activities of individuals related to system operation Document handling Informal communications	Observer may have biases or influence processes observed. Limited to a small number of activities. Observation skills are not easily learned.
Questionnaires	Written responses to a set of questions from a large number of persons	Questionnaire design can influence the validity and worth of information collected.Questionnaire responses must be brief, easily recorded, and unambiguous.A follow-up can increase returns.
Interviews	Oral responses to questions from a selected sample of persons	Sampling allows quicker and more economical collection of data. Interview must be carefully planned and skillfully executed.As a follow-up, interviewee should have opportunity to review and comment on written interview summary.

2.3.1 *Requirements Collection*

Before collecting requirements, a plan is prepared to determine the data to collect the potential sources of data. A schedule of activities is also prepared. Overall organization system plan(s), the underlying (requirements) models and organizational practices influence requirements collection significantly.

The activity is allocated only enough resources for successful completion; due to such a restric-

Table 2
Manual techniques for specification and analysis

Technique	Description	Tools/Techniques
ARDI	System development is described as four phases: Analysis; Requirements determination; Design and development; Implementation and evaluation	PERT charts; Documentation standards for each phase; Project Management Technique
Hierarchical Input Process Output (HIPO)	Graphical design aid and documentation technique. Decomposed system is documented in terms of inputs, processes, and outputs.	Charts: Visual Table of Contents; Detailed HIPO diagrams; Coding pads, templates, manuals
Structural Analysis and Design Technique (SADT)	Techniques for top-down, structured approach to requirements documentation, project planning, managing and evaluation	Graphic documentation technique: activity and data diagrams; Definition of personnel roles
Data Flow Diagrams/ Structured Analysis (SA)	Top-down techniques, using directed graphs, in which processing and information requirements are integrated and collected simultaneously.	Data flow diagrams

tion, all requirement data may not be collected, nor all sources contacted. Trade-off must be made by the coordinator. The effectiveness of this activity is highly dependent on the collectors' perspective. They must concentrate on determining *what* the ensuing system should do, not on *how* this should be accomplished.

There are four major data collection techniques: (1) Reviewing documentation, (2) Observing the operating environment, (3) Administering questionnaires, and (4) Interviewing pertinent individuals. Table 1 summarizes these four techniques.

2.3.2. Requirements Specification and Analysis

Requirements specification is the activity of documenting the requirements in a precise and standard form. Requirement analysis is the activity of determining the completeness, consistency, correctness and validity of the requirement docu-

Table 3
Computer-Aided Techniques for Specification and Analysis

Technique	Description	Computerized Features
CADES	Uses structural modeling technique to generate an information-oriented system design. Generates implementation code and test data from design specification.	Design-information database; Implementation code database; Automatic generation of code and test data.
PSL/PSA	Provides user with PSL, a structural language, to document *what* system requirements are, PSA programs are used to create, analyze and generate report from the data base of PSL specifications.	(All in PSA) Verify syntax and semantics of PSL statements. Create, modify and access design data base. Produce a variety of standard reports utilizing four modes of presentation: lists and tables, matrices, pictures, and text.
IDEF	Uses forms-driven graphical technique to specify entity-relationship-attribute (ERA) models. Analysis enhanced through defined roles of analysis team members and specific analysis procedures.	ERA dictionary database and reporting programs

ments. The type(s) and method(s) of analysis is dependent on the method(s) of specification. The technique may be manual or computer-aided.

2.3.2.1 Manual Techniques. In manual methods, the analyst collects the data required, collates, continually organizes and analyzes the data, and then produces reports. The data is documented in a combination of text, tables, diagrams, flowcharts and decision tables.

There are basically two categories; the first is text oriented, utilizing natural and formal language and charts. Examples are the AUXCO method [5], NCR's ADS, and ARDI [27,48]. The second uses graphically oriented techniques to document pictorially; text is used sparingly to augment the diagrams. Examples are HIPO [58,61]; SADT [88,89,90]; and DFD [38]. The characteristics are summarized in Table 2.

2.3.2.2 Computer-Aided Techniques. There are two categories of computer-aided techniques. In the first, the computer facilitates the use of an existing manual method. In the second, new techniques are designed to make use of the computer. This has three basic components:

a) A requirements statement language that is sufficiently structured for computer processing and analysis.
b) A software package to store, analyze, retrieve and display its information.
c) A database to store the requirements and facilitate analysis by the analyst.

There are a number of computer-aided requirements techniques: Computer-Aided Design of Information Systems (CADIS of the Royal Institute of Technology in Stockholm) [17]; Computer-Aided Systems Construction and Documentation Environment (CASCADE of the University of Trondheim) [1]; Computer-Aided Design and Evaluation System (CADES of International Computer Limited) [54]; Problem Statement Language/Problem Statement Analyzer (PSL/PSA of the ISDOS Project at the University of Michigan) [56,57]; and IDEF (ICAM Definition Method developed for the U.S. Air Force by Softech, Inc. and Hughes Aircraft Company) [14]. Their characteristics are summarized in Table 3.

When any organization is considering a manual or computer-aided requirement technique, the costs and benefits should be considered. The costs include technique acquisition, computer costs (installation and operation), personnel, personnel training, etc. The benefits of computer aids may include better requirements documents, more user involvement, improved coordination of the activities, better access to requirements, and improved system development with reduced time and costs.

2.4. *Future of Requirements Collection and Analysis*

Progress in requirements collection and analysis will depend largely on the development of precise, generally accepted terminology, the understanding of goals during requirements analysis, and availability of tools to manage the collection and manipulation of requirements.

2.4.1. Standard Terminology

The development of a standard terminology is extremely important; e.g., "requirements" and "specifications" are sometimes used synonymously while in some instances they describe general and detailed documents, respectively. Clearly, such differences are apt to lead to confusion and errors.

2.4.2. Desirable Goals of Future Methodologies

A comprehensive methodology must specify precisely what outputs are to be produced at each stage. The methodology must describe how quality is to be verified and maintained. In the requirements phase, this is particularly important, because of difficulties in communicating with users, its subjective nature and high cost of errors found in later phases. The methodology should be adaptable: it should handle simple or complex corporate structures and information system structures, sophisticated or naive users, minor revisions to existing procedures or completely new systems, etc. The methodology should be hierarchically structured, with the degree of details selected to conform to the need.

Productive use will also require that the techniques be much easier to use. In addition, there are already problems in personnel training. It is desirable to develop very simple techniques and models for direct operation by the user. Menu-driven, question-and-answer, and other methods are generally easy to use and provide feedback more quickly and reliably than a systems analyst. Moreover, the people who recognize errors – the users, designers and managers – should all be involved in the design and implementation from the requirements phase. Planning in present efforts seems rarely adequate, either because its importance is unrecognized, or because it has a high-visibility cost with no easily quantifiable benefit, or because the tools and methodology for adequate planning are unknown or nonexistent.

2.4.3 Desirable Goals in Requirements Collection

To improve the quality of the requirement collection effort, the following factors should be observed:

i) The analysts should be trained appropriately.
ii) Requirements collection activity should be kept independent of the later system details such as data models, type of software and hardware, etc.
iii) Development of standards for naming and user of computer-aided techniques to cope with a large number of names.
iv) Development of a methodology for recognizing and correcting errors during requirement collection.

Discussion

A requirement collection analyst should be selected on the basis of psychological profiles, aptitude tests and an evaluation of previous experience. The analyst should have a high degree of interpersonal skill and tact in dealing with nontechnical people. Analysts should also be trained to accept different viewpoints.

Since requirements collection requires one to sift through volumes of data, a lot of which may be irrelevant. It is tempting to look toward a certain end result such as a particular data model or a specific system and collect requirements relevant to only that target. Problems crop up when the model or system is changed. It is essential to carry out the requirements phase independently of specific systems or models.

Naming problems may cause costly iterations of requirement collection. Different names for the same concept (synonyms) or same name for different concepts (homonyms) should be recognized and resolved as early as possible.

2.4.4 Goals in Requirements Specification and Analysis

The following goals are particular to the specification and analysis of requirements:

i) *Development of "Common Sense" Analyses*: "Common sense" analyses are generally not sophisticated but are extremely valuable. They may be used to suggest requirements which the user has failed to express because they are implicit or too obvious to be recognized consciously. They are learned through experience and are generally not well documented. Two types of common sense analyses are particularly important: heuristics or rules of thumb based on human factors, business principles, etc., and the deduction of consequences of requirements. The heuristics may suggest that a user is asking for too much data, or is imposing unnecessarily short response time, or is ignoring an important segment of the application. Requirements should not be made too restrictive or too liberal; the analyst should be able to judge its consequences.

ii) *Techniques for comparing the results of phases*: For example, the Enterprise Requirements Model could be compared with an application area requirements model to ensure that there is no conflict. This appears to require the development of rather complex mappings.

iii) *Developing techniques to render the models flexible and evolvable*: Another goal is that of developing techniques for determining flexibility or "robustness" of long range Business Plan, the Enterprise Requirements Model, and the Global Information Model. These models always evolve over time. It would be very desirable to have some objective measure of the degree of flexibility achieved, and the causes of the lack of flexibility.

iv) *Making changes and propagating them*: A final goal is the development of a mechanism for easily making and propagating changes within a particular model and propagating changes between models. This capability requires the previously mentioned mappings between models, and probably a man-machine dialogue.

2.5. The Role of the Data Dictionary System in Requirements Analysis

Data dictionary systems can play a central role in the requirements phase. DDS can relieve the clerical burden of the collection activity. In the specification phase, the DDS can serve both as a tool and as a control mechanism. Finally, the dictionary can provide certain basic types of analyses and also serve as a database to which more sophisticated analytical tools may be applied.

2.5.1 Collection

Determination of requirements is basically a human activity in which the role of the DDS is one of a passive repository for the documentation of requirements. The amount of information gathered initially may be quite voluminous. Use of the DDS can reduce the burden of compiling and cross-referencing this information manually. Further, if proper dictionary entry types, e.g. data elements, reports,, screens, etc., are available, the use of the DDS can have a standardizing effect on the efforts of several analysts, or of one analyst surveying several users.

If the data dictionary system is used as a tool for documenting the enterprise model and existing applications, analysts can use the dictionary database for reference and direction during the collection phase. Data sources can be identified, existing databases can be located and may be determined from the available dictionary information.

The DDS characteristics most necessary to support an analyst during collection is ease of use, both for recording newly collected data and for accessing existing documentation. The inquiry/reporting capability of the DDS must be powerful enough to support the analyst.

2.5.2 Specification

The constructs and rules of the requirements specification methodology must be designed to force the analyst to completely and unambiguously specify the data and processes required. The analyst typically reviews the specifications with users, subjects the requirements specifications to one's own analyses (the analysis phase) and recreates the specifications as a result. When complete, the specifications should provide a full picture of user requirements and should contain enough detail for logical database design.

The major contribution that a data dictionary system can make to specification is to support the primitives, e.g. object types and relationships, of the methodology being employed by the analyst for specification. For example, if the analyst is using some variant of the E-R model [22] as a specification tool, the DDS should be capable of supporting objects such as entities, entity sets, attributes, relationships, relationship sets, cardinalities of entity sets, minimum and maximum cardinalities of relationship sets, etc. In addition to enabling the analyst to record characteristics of interest about these primitives, the DDS should also be able to exercise control over the capture of this information. For example, if 'NAME' and 'KEY' and 'CARDINALITY' are three important attributes of object 'ENTITY', then the DDS should reject any instance of ENTITY in which these attributes are not specified.

The dictionary system should support a variety of output modes, e.g. graphics, formatted text, tables, etc., so that the analyst may select the form most appropriate to the users. Basic controls on names of objects and relationships, can identify redundancies and inconsistencies. Controls, specific to the methodology itself, are also desirable. For example, in Structured Analysis (SA) [38], each dataflow must have a source process and a destination process. The DDS should check and enforce this requirement.

2.5.3. Analysis

In requirement analysis the analyst critically reviews the specification of requirements in order to reduce redundancy, ensure consistency, and examine the interfaces between existing requirements/systems and the newly specified data needs. Questions of semantics and completeness must be raised and resolved.

System analysis is a human mental activity rather than a programmed task. The data dictionary system can serve best as an aid to the analyst rather than as a substitute for the analyst. As a minimum the DDS must be capable of generating crossreference reports on the objects defined in the dictionary. Where-used reports and traces of the effects of change are also useful. It should be possible to geneiate reports by object types (e.g., report of all entities), by temporal attributes (e.g., a report of all weekly processes) or by specific (attribute, value) combinations, e.g., (for item ='EMP-NAME': show ...).

Completeness checks can be performed easily by a DDS facility. All definitions can be cross-checked and any standards or consistency rules posited by the specification methodology can be checked and verified.

2.5.4. The Development of the Enterprise Model

To assist in the development of an Enterprise

Model, involving the same phases as application requirements analysis, the DDS must support a variety of constructs – business processes, organizational units, organizational activities, and data groupings – relevant to the above processes and activities. DDS must be able to accommodate the handling and recording of the evaluation of the above aspects. Appropriate listings, cross reference reports must be supported.

To support both enterprise and application modeling, the DDS must be able to record information on objects at several levels of abstraction. For example, one must be able to represent the data class EMPLOYEE, the subclass SECRETARY, and the record type EMPLOYEE RECORD. The DDS must be able to document the differences between these objects at different levels as well as the mapping(s) necessary to go from one level to another. Finally, the DDS must have an ability to check for consistency between levels as well as within levels. For example, if the above named EMPLOYEE RECORD is defined as a part of the Secretarial Skills Inventory Application, which of the attributes of EMPLOYEE belong to the subclass SECRETARY?

2.5.5 Summary

There are three categories of features required for a data dictionary system to support the activities of requirements analysis: definition, access (reporting), and control (see Table 4). Unfortunately, most data dictionary systems have been developed with support for DBMS data definitions as the primary objective. Currently, none provide the full range of flexibility and analytic capability necessary for the requirements analysis task. Existing packages include, at best, some reporting features, limited support for non-standard object definitions, and some basic controls such as checking for duplicate object names.

Further research and development is required on more flexible, extensive and extensible definitional capabilities, on user-oriented modes of output (such as graphics), and on more extensive, perhaps user-specified, controls.

One recent proposal for an Information Resource Dictionary System (IRDS) is that of Kerschberg, Marchand, and Sen [63]. The IRDS has as its data model the Enterprise Model, which looks upon an enterprise as a collection of business functions. Each of these functions, in turn, consists of a collection of activities with components at the three levels of management – operational, tactical, and strategic. Activities are specified as collections of *views* which incorporate three aspects: 1) organizational dynamics specified in terms of events, states, and their temporal and logical interrelationships, 2) forms, form operations and form routings, and 3) data requirements expressed in terms of Functional Data

Table 4
Data dictionary system characteristics and capabilities needed to support the requirements phase

Characteristics/Capabilities	Relevant Task(s) in Requirements Phase			
	Collection	Specification	Analysis	Enterprise Modeling
Definition:				
Support for a variety of information and processing model(s) primitives.	×	×		
Support for several levels of abstraction	×			×
Tailoring to a specific system development methodology	×	×		
Access:				
Ease of use	×			×
Selectivity of output requests	×		×	
Cross-reference and where-used reports		×	×	
Variety of output modes		×		×
Control:				
Enforcement of naming standards	×		×	
Verification/enforcement of methodology-related rules	×		×	
Controlled evolution of definitions	×	×		×

Model data abstractions. Each view is performed at a particular workstation.

The view mechanism is an aggregation of three distinct yet interrelated components: events, forms, and data. Events trigger actions which in turn operate on forms. Forms, in turn, may be routed from station to station (message passing) as a means of coordinating activities associated with a business function. In addition, forms have attributes which take on data values as they flow through the organization; the forms are data abstractions which are a source of data requirements for database support.

This modeling approach leads to new ways of thinking about entities, their properties, and how they relate to other entities. For example, entities "pick up" property values as they are operated upon by the "real-world" – in fact they have a temporal history of the entity. The view mechanism allows the modeler to encapsulate a collection of events, states, forms and operations. The flow of forms allows views to communicate and request service from other views.

If one carries this line of thinking a bit further, it becomes obvious that "intelligence" concerning the application – the collection of views modeling a business function – can be distributed in many ways. An entity can know everything that could happen to it, including exceptional cases. Forms can aid users by having built-in "help" information for each form attribute. Form procedures can be attached to forms and would only be activated if the form had the proper information and were in the proper "state."

During logical design the enterprise views are integrated into a global database schema. Event-oriented information is mapped to time-stamp attributes of the relevant entity types. Forms are mapped into entity types and the views themselves are mapped to aggregations of entity types.

3. Information Modeling

3.1 The Information Modeling Process

The information modeling process takes as input the enterprise requirements specification. The goal of information modeling is to obtain an integrated, formal implementation-dependent specification (the information schema) of application-specific enterprise information.

The steps of the database design process are summarized in Table 5. Steps 4 through 7 correspond to information modeling.

The specification is interested in that it is the product of the view integration process wherein the requirements of functional organizational units are reconciled and integrated into the global information schema. The specification is formal in that the structural and operational semantic of the information model are precise and well understood. The information model should represent enterprise concepts, structures, operations, and constraints. Lastly, the information model is implementation independent in that mappings should be provided to map the information schema into a database management system (e.g., the relational model [24,25] as implemented in System R [3], INGRES [103], the CODASYL model as implemented in IDMS [28] or SEED [42], or the hierarchical model as implemented on IMS [52] or System 2000 [76]).

3.1.1. Desirable Features of an Informational Model

First and foremost, an information model should provide a collection of semantic constructs to aid the database administrator in modeling the database-specific portion of the enterprise model that results from the requirements analysis phase.

Semantic constructs include the *structures*, *operations*, and *constraints* available to specify the semantics of the enterprise as it relates to the database schema that will support user applications. A minimal requirement is that the information model support the notions of aggregation and generalization [100,73]. Further, the model should have a sound mathematical foundation as well as a collection of "orthogonal" concepts, that is, the modeling constructs should be free of overlap so that a real world concept will have a natural and unique representation in the model.

Most existing models support the notions of data type (domains), entity sets, associations, properties of both entities and associations, and subtyping (generalization). Some models include the notion of time with the even concept. Some may include time at the enterprise level where events and procedures [109] are specified. These events and procedures map to triggers and transactions at the information model level.

Since the role of the information model is to represent user applications and integrate them into

Table 5
Action Table for Database Design *

SPECTRUM			
User		Database Administration	Technical support
		1: Identify user groups.	
2: Collect requirements.			
	3: Structure requirements into enterprise model and application models.		
		4: Integrate application models into an information model.	
5: Identify data ownership.			
	6: Correct models.		
		7: Make integrated database schema available for review.	
8: Define database subschemas based on knowledge gained during integration.			
			9: Propose implementation alternatives (DBMS types or file-based alternatives).
10: Apply transactions to the subschemas and estimate relation and connection access frequencies.			
		11: Compute aggregate load applied to the integrated database schema.	
			12. Design promising inplementations in detail.
13: Identify response time constraints			
		14: Apply critical functions to the schema for response time assessment.	
			15. Compute aggregate performance and cost factors.
		16. Select a design that appears viable.	
			17. Compute performance where response time is critical.
18: Check if selection and response times found satisfactory.			
	19. — — — n: Iterate !		
User		Database Administration	Technical support
SPECTRUM			

a global conceptual schema, the model should provide a view integration mechanisms. Moreover, to map the information model effectively to logical and physical database structures, the specification of processing requirements is essential. Both view integration and process specification will be discussed shortly.

Finally, the information model should be implementation independent in that its concepts address the modeling of the "real world" rather than being directed toward the logical or physical structures of a target database management system. Several types of mappings are associated with the information model: mappings to user views (external schemas), logical structures, and physical structures.

3.1.1.1. View Integration. Database design being an extremely complex task, it is essential that the problem be subdivided into manageable parts to reduce complexity and facilitate staged implementation. Usually, such partitioning is performed by

This table is supplied by Gio Wiederhold.

application areas (e.g., accounting, marketing, finance, etc.) giving rise to the notion of "local views." The view integration process consists of combining local views into a consistent global view (conceptual schema). Combining these local views involves resolving name conflicts (synonyms and homonyms), removing redundant relationships, combining entities, and identifying and resolving insertion, deletion, and modification anomalies.

On the one hand, models based on mathematical functions are well suited for view integration. This is because all data are represented in their most elementary form and are not biased toward a particular design. However, the mathematical integration procedures involve a synthesis of functional dependencies which are extremely hard to enumerate exhaustively. Therefore, researchers like Navathe and El Masri [30, 32, 81, 85], Batini and Lenzerini [4,7,8], have been advocating an object integration approach. Also, while the view integration process is conceptually simple in that it involves the combining of subgraphs according to a set of formal rules and heuristics, in practice it is very difficult to accomplish.

View integration must be considered an interactive process since the designer must be constantly prompted for additional information. Facilities must be provided to allow the designer to merge nodes, remove redundant functions, and input additional assertions. Of course, the data dictionary system is required to store the information and keep track of all design decisions.

3.1.1.2. Process Requirement Specification Support. Processing information is needed to ensure correctness and efficiency of the design. Approaches that are based solely on semantic structure information are unable to estimate the processing requirements of a design. Conversely, however, designs based strictly on processing considerations tend to be flexible. Modeling of processing in the context of a well formed semantic model is useful for:

- Determining inconsistencies in the integrity assertions (i.e., update operations are found to violate the integrity rules stated in the information model).
- Determining relative access frequencies along functional paths. This information is useful to the physical design phase.
- Identifying deficiencies in the semantic model. Given a large number of entity types, there are a vast number of potential nonfunctional relationships (e.g., many-to-many, n-ary relationships) and subtypes (or generalizations) that could be defined. The exercise of modeling the key processes aids in identifying additional schema constructs that are needed, and existing relationships in the model that are nonessential.
- Providing formal specification for detailed application development. After the design has been approved, the process specification serves as a specification for application software developers.

For any process modeling language to be useful in database design, it is essential that it provide constructs that obviate irrelevant details (with respect to logical database design). Furthermore, varying degrees of procedurality should be allowed to accommodate different levels of modeling (e.g., a high-level specification of processing intent, versus procedural navigation through the schema).

3.2.A Synopsis of Information Models

The goal of the information model is to represent the real world enterprise specification in terms of the structures, operations and constraints of a data model. The traditional data models (e.g., relational, network, and hierarchical) lack many of the modeling constructs necessary to model the real-world objects, properties, activities, etc.

Thus, over the last several years, there has been an intense research effort to develop *semantic data models* that capture the "meaning" of the real world, specify these concepts, and communicate that meaning to Database Administrators. An important goal is to have end-users understand the real world representation; but at present this is still a distant goal.

Rather than provide descriptions of each model surveyed, we have decided to provide a *taxonomy* based on important concepts and features common to these models.

The taxonomy framework looks at the conceptual structures, operations constraints, and specifications of the semantic data model.

Conceptual Structures – A model must represent both the concrete and abstract objects that are found in the enterprise: entities, properties of entities, relationships among entities, temporal and logical relationships, etc. The notions of aggregations and generalization are well-accepted abstraction techniques to allow the modeling of complex data organizations. Aggregation is used to group objects, properties, and even relationships into new concepts. Generalization (and its converse, specialization) allows the model builder to *generalize* several diverse concepts to a new concept that focuses on their common properties. For example, Engineers, Secretaries, and Machinists are all Employees of a company. Properties common to all Employees might be: date-of-hire, date-of-birth, salary, etc. However, such properties as typing speed, engineering degree, and machine-types refer specifically to Secretaries, Engineers and Machinists, respectively.

Operations – Once an enterprise has been represented by an information model, the resulting description is called an information schema. The schema governs the behavior of the actual *instances* in the populated database. In order to populate the database and modify values to reflect real world conditions, the model (and the system that implements the model) must be able to handle simple insert, delete and update operations, as well as more complex transactions.

The operations must ensure the "semantic integrity" of the database by observing the constraints implicitly in the structures and explicitly specified by assertions (to be discussed shortly). The problem of semantic integrity enforcement deals with the propagation of constraints (other operations) from concept to concept.

Thus, a simple delete operation may entail the propagation of other deletions. For example, the cancellation of a project not only deletes the project-object from the database but also implies the reassignment of some employees and perhaps even the firing of others. It may also cause updates to budgeting information, inventory information, etc.

Constraints – Semantic data models have many constraints implied by their structures. Other constraints may be explicitly stated by assertions.

For objects we might specify that an object exists in relation to the existence of another object or relationship, e.g., the dependents of an Employee or the spouse (by marriage) of the Employee.

The properties of objects may be constrained in the cardinality of their values, whether they are keys (i.e., they uniquely identify an object), and in terms of their mapping type (e.g., 1 : 1, onto, total, and partial).

Constraints on associations may refer to functional and multivalued dependencies among objects, and special semantics associated with operations.

In the creation of subtype/supertype hierarchies via generalization, constraints express whether the subtypes *Partition* the supertype into disjoint subsets. For example, consider the constraint: "Employees *must* be exclusively Engineers, Secretaries, or Machinists." This constraint would not only partition the set of Employees, but it would also imply that the enterprise would have to assign a new employee to one of the subtypes.

Another constraint associated with generalization/specialization is the *cover* constraint which states that the union of the subtype entities contains all the entities in the supertype. However, it is possible to have a single entity belong to more than one subtype.

Specification Languages – refer to those facilities that allow the user to *specify* an information schema by using a Data Definition Language, to *manipulate* the populated database through a Data Manipulation Language, and to *query* the database by means of a Query Language.

Another important facility of an information model is its ability to query and update the schema that is being designed. Since the construction of a global information schema requires the *integration* of local views, we feel that a good semantic data model should provide such facilities.

The Models Surveyed

The semantic data models that are candidates for the *information model* are:

1) The Functional Data Model [15,64,65,96,98, 99,113]
2) The RM/T (Relational Model/Tasmania) [25,49]
3) The Semantic Hierarchy Model [100]
4) The Entity/Relationship Model [22,91]
5) The Semantic Data Model [46,71,72]
6) The Navathe/Schkolnick Model [82,85]
7) The Structural Model [33]
8) The Object/Role Model [34,83], and
9) The Semantic Association Model [104,106].

Table 6 summarizes the features of the models with respect to the taxonomy framework. Clearly, many models have comparable concepts. The issue is not which model is better. The real issue in which model will become (commercially) available in the near future. It has been reported [23] that currently several of the above models are actually being used in organizations to model information. The Entity-Relationship Model and its variants, by far, are the most popular models. The development of such models will greatly benefit the database design process.

3.3. Tools for Database Design

Once an organization decides that data is a resource to be managed, and adopts database management system technology, it is faced with the problem of effectively *managing* the database system life cycle. The life cycle [108] can be divided into two phases: 1) *application development* consisting of requirements analysis and specification, logical design and physical design, and 2) *database operation* consisting of implementation, operation, tuning, and adaptation.

A *database workbench* environment is needed to specify, design, develop, test, and tune database-intensive applications. The workbench should be viewed as a collection of independent yet communicating tools. Each tool should be designed for a specific part of the life cycle and should have the capability of communicating with other tools. This communication is important in supporting the mappings between the levels of the database design process depicted in Figure 1. The workbench would rely on a data dictionary system for inter-tool communication. Information common to all tools would be stored in the dictionary and could be used by the tools as well as by the database designers.

Several database design tools have been experimentally built in research settings. Some examples are:

i) The logical and physical design tools of Katz and Wong [59,60] from UC-Berkeley based on a design model similar to the entity relationship model.
ii) A collection of tools from University of Pennsylvania for the design of CODASYL databases [41,42,43]. DESIGNER produces integrated data structures diagrams from a formal specification of requirements in the form of queries. DBD-DSS is a decision support system for physical database design whose output can become the input to the File Definition Processor of the DBMS SEED [42]. A dynamic restructing tool with limited capabilities is also proposed.
iii) The Conceptual Data Model, the language TASL and accompanying design tools [113] from Purdue University.
iv) A physical database design tool called DBDSGN from IBM Research [92] which works in conjunction with System R.
v) The INtegrated Database Design Aid from the University of Michigan which was postulated as an integrated set of tools [35,55,75] for creating a variety of database designs and experimenting with different design strategies.

A major project, "Data DEsign Workbench" (DDEW) at Computer Corporation of America [87] is attempting to provide an extensive set of graphic facilities to the database administrators and designers to cope with the design problem. The emphasis is on the ease of interface rather than on elaborate optimal design algorithms.

Another project at Honeywell Corporate Research Center [30,31] is designing tools for schema integration which encompasses the view integration in logical database design and database integration in a distributed database environment. Their emphasis is on algorithms and mappings.

A major project in Italy with the cooperation of the Italian Government and the National Research Council (CNR) was active during 1980–85. This DATAID project has resulted in the most comprehensive methodology for database design so far; a collection of papers is compiled in [19]. A number of independent tools which can interface with one another have been built and reported in [2].

Commercially, very few database design tools are available. Tools such as Data Designer [29] have very crude user interfaces. This particular tool collects all the dependencies among data and proposes a set of third normal form relations [24] which covers only a small part of the database design problem.

3.3.1. The Role of the Data Dictionary System in Integrating Logical Database Design Tools

The data dictionary system is viewed as playing an *active* and *central* role as the repository of the information needed by the workbench tools. Moreover, it is the mechanism by which tools

Table 6
Features of semantic data models

Features	Models								
	Functional Model	RM/T	Sematic Hierarchy Model	Entity Relationship	Semantic Data Model	Navathe/ Schkolnick	Structural	Object/ Role	Semantic Association
Structures									
object (concrete)	entity	surrogate kernel entity	object	entity	class/object	object	entity	object	atomic concept
object (abstract)	entity, event	events, scripts	object hierarchies	relationship	class/event hierarchy			event	non-atomic concept
properties	function (single and set-valued)	property characteristic (multi-valued)	attributes	attributes	attributes characteris-tics	attribute	attribute/ domain	significa-tions/roles	via member-ship asso-ciation
associations	associative entity	associative entity, cover aggregation	aggregation	relationships, relationship hierarchies	aggregation cover aggre-gation	identifying association	relation-ships	association (hierarchies)	membership aggregation interaction
generalization/ specialization	ISA-function	ISA-hierarchies	generalization	ISA-hierarchies	generaliza-tion	simple association (categoriza-tion, selec-tion, sub-setting)	–	–	generalization (also composi-tition, cross-product, sum-marization)
Operations									
insert, delete, update	yes	yes	yes	yes	yes	yes	yes	-	yes
transactions	yes (TASL)	yes (event scripts)	yes	–	yes	–	–	–	–

Constraints on									
objects	typing (in function)	typing	typing, existence dependency	existence dependency	yes	identification insertion deletion	primary (existence) cardinality	existence	yes
properties	key, partial, total, 1 : 1; onto, cardinal-ity	property integrity (recursive)	functional dependency, no null values	key, cardinality	yes	total and partial identifiers	Lexicon, referenced, nest (existence dependency)	key, integ-rity, car-dinality	yes
associations	functional dependencies, referential integrity	referential integrity	referential integrity	key, weak entity	onto, total/partial	same as objects	association	referential integrity	yes
generalization/ specialization	partition, cover	partition, cover	partition	–	partition, cover	partition, complex conditions	–	–	yes
Languages									
Data Definition	DAPLEX	yes	yes	yes	yes	yes	–	yes	yes
Data Manipulation	DAPLEX, FQL	yes	yes	yes	yes	yes	–	yes	yes
Query	DAPLEX, FQL, SYNGLISH	Relational Algebra & Calc.	–	yes	yes	–	– yes	yes	–
View Integration	yes	–	–	–	yes	yes	yes	yes	–

communicate with one another. Thus, the DDS has as one of its components a *meta-database* about the database being designed.

We envision an expanded role for the data dictionary system, as compared with commercially available products. In addition to providing lexicons of data item definition, data typing, and synonyms, the dictionary system concept should be extended to provide the following services:

Design Log. To effectively document the decisions made at the various design stages, the dictionary system must have a logging facility. This feature would record the various options considered and state the reasons for particular decisions. For example, user processing requirements on an entity set might dictate a subtype partitioning of that set. This fact should be recorded and be available for future reference should processing requirements change.

Design Audit Trail. The complexity of database design leads to the decomposition of the design process into manageable phases, e.g., requirements analysis, enterprise modeling, information modeling and view interaction, logical and physical structure design, and performance tuning. The phases are linked, however, by the mappings that transform objects, operations and constraints from one phase to the next. The dictionary system should provide an audit trail that records the mappings
used together with relevant parameter values. This information could be used to characterize "workload" from level to level. Sensitivity of a design to the parameters available to the designer should be determined via sensitivity analysis. The audit trail feature also allows high-level user views to be related to the application processes that manipulate database objects. Inverse mappings are also important, namely, given a database object, what processes act on it, from which user view(s), and with what frequency. Information of this type is crucial for view integration and for physical design.

Inter-tool Communication. Workbench tools must be able to communicate with one another. For example, the constraints specified in the information model should be made available to other tools such as a form system for consistency checking of input data, application development aids which ensure the semantic integrity of user controlled updates, query optimizers that use access path frequencies, etc.

Metadata Management. The features mentioned above show that the prime function of our data dictionary system is the management of metadata. The importance of this function is addressed in the next section.

3.3.2. The Role of Metadata in the Data Dictionary System

3.3.2.1. Why Metadata (?). The advantages of having metadata accessible to the user are almost self-evident. In an environment containing several independent and accessible databases, a user may simply want to know in which database or file a given lexical descriptor is meaningful. Since the internal descriptors are often coded or highly abbreviated, and since the user may initially use a term different from that used in defining the database, some kind of lexical translation will be needed. Textual comments are also important.

Metadata should also describe structural relationships and should make them apparent to the user. A query issued in a high-level query language based on a misconception of structural relationships can often produce misleading results. For example, a relationship may be one-many in the database while the user thinks that it is one-one; there may be no direct relationship between two entities when the user thinks there is one.

At a lower level, data formats, report formats and the characteristics of system interfaces, are also metadata that are useful to those who wish to affect data transfer between various systems.

Finally, metadata may also be exploited by other programs. High-level query systems and natural language systems both exploit metadata for navigational purposes within a database. It is usually the case that this metadata must be manually constructed for these systems, simply because it is not held in a clean structural form either within the database or as an adjunct to it.

3.3.2.2. Querying Metadata. Many interactive query systems allow for some form of querying of metadata as well as of data. It is desirable that the query language for metadata resemble the regular query language. However, for this to be successful, some representation of the metadata must be found within the data model itself. The existing commonly used data models (relational, network, etc.) are not rich enough to permit this and a more sophisticated "semantic" model is required before metadata may be concisely embedded within the database itself.

The complexity of metadata depends on the purpose for which it is used. For help in formulating queries, a relatively simple model that informs the user of the terminology of the database, some informal description of the contents, and some more formal description of the relationships within the database (functional, many-many, etc.) is sufficient. For the purposes of updating and understanding integrity constraints, the vehicles and models by which users can query a DDS would be much more sophisticated.

3.3.2.3. Operations of Metadata. This is one of the most difficult problems associated with the use of metadata. Should it be possible to operate upon metadata in order to update it or to produce new metadata? Some useful techniques may be conveniently described as operations on metadata, e.g., the generation of a user view. Accompanying this mapping there must be an associated transformation of database access primitives. If user views are to be constructed by operating upon metadata, the primitive operators should be well defined, and some kind of calculus or algebra should be available for metadata operations.

The operators needed to construct user views are those that "lose" information from the database. More complicated (and even less well-understood) are operators that add information. There are a number of techniques that would also benefit from being formally presented as operations on metadata. They include constructing "superviews" [16]: a view that integrates data in several independent databases; the related problem of merging databases; and the problem of dynamically extending the schema of an existing database to include new data. These are all problems that will require extensive research in database semantics for their resolution.

3.3.4. Desirable Workbench Tools

The following list represents a possible categorization of the workbench tools.

3.3.4.1. Information Models as Tools. Existing information models provide semantic constructs, operations, and intrinsic constraints required to model user and global information schemas. Although none of these models is machine-implemented at this time, we expect to see several operational in the next few years. Needless to say, the information model provides metadata to the data dictionary system, and is therefore of utmost importance to the workbench concept. The model should have mechanisms to support both view integration and process specification.

3.3.4.2. Mapping Tools. Support tools are needed for the various mappings in the design process: enterprise model (the result of requirements analysis) to information model, information model to a logical database schema, and conversion of logical schemas into physical schemas.

3.3.4.3. Constraint Subsystem. The constraints defined in the Integrated Conceptual Database model can be implemented in four ways:

a. Some constraints are supported by the database implementation structure, i.e., by the a *priori* behavior of the data modeling constructs (e.g., ownership constraints are enforced by implementation hierarchies).

b. The remaining constraints may be stored as assertions, for interpretation at query process time [98].

c. Assertions may be automatically bound into access procedures during compilation phases that bind external schemas to the internal representation. Some experiments on automatic constraint procedure generation have been made, but no consistent theory exists.

d. If no assertion subsystem is available, then the constraints should be included in procedural interfaces that are implicitly or explicitly invoked by programs using the database.

3.3.4.4. Tools to supply "Run-time" information. Knowledge stored in the conceptual model can be profitably used in the processing of queries. For example, in a relational database the model should document all enforced 1 : N connections between relations so that the best join method can be chosen. The model must also document alternate paths to the result, so that the best access paths among logically equivalent paths can be determined.

3.3.4.5. Application Program Development Aids. The knowledge embodied in the schema can substantially reduce the effort during the design and planning phase for data acquisition for application programs. Furthermore, if the constraints can be used to automatically generate modules which check range, maintain the properties of relationships, update derived data, etc., then the coding effort is reduced. But more importantly, the reliability of the database and the relevance of its contents will be greatly enhanced.

3.4. Database System Communication

The need for such communication may arise for two reasons:

(1) user requirements for local and shared ownership and control of information may dictate a "distribution" of the information model, a different "data model" for each application area, and possibly the use of different DBMS and hardware configurations, and

(2) the systems may have been developed independently, may be operational and must be linked together because of the overall enterprise information needs.

3.4.1. Organizational Factors of Distributed Databases

When enterprises move into database-oriented processing, an apparent conflict arises. The centralized collection of information implied by the database technology is often seen as the property of the enterprise. This notion is in conflict with organizational principles of delegation of responsibility and authority. Ownership of information is normally associated with power and the ability to act but excessive centralization limits initiative and may sometimes fail to realize the full potential of information or lead to its misuse. Most organizations attempt a logical decentralization of the database, possibly coupled with physical distribution.

Data *ownership* describes a user's relationship to the data in terms of responsibility for data and authority for access to data. If, for example, a bank branch manager is responsible for the bank's customers in the local area, then maintenance of the customer lists, their addresses, and the linkage to various accounts is the responsibility of this manager and in that sense he is the owner of that data. The integrated schema may impose some constraints, e.g., a customer with a non-zero balance may not be deleted. However, it must support all authorized actions, such as the entry of a new customer. The integrated schema provides the crucial information needed to coordinate the activities of the local schemas. Data ownership is noted in the information model. All data described must either be owned or be generatable from other data through the use of some computational procedure. These procedures are attached to the information model.

When the frequency of retrieval of locally owned data becomes very high at remote sites, performance demands may dictate replication, i.e., storage of extra copies at those sites. For a retrieval-only system, each data object may be stored in terms of one master copy at a site with several secondary copies at selected other sites. Master copies, together with secondary copies of data may then be periodically updated. Systems allowing on line updates and keeping replicated data have to adopt more complicated mechanisms to keep all copies "current". The notion of a true-copy token [74] which identifies the primary version may be used to control such databases. The true-copy token resides where the update privilege resides, and is moved as needed. Before it is moved, inter-site consistency is established. The movement of the true-copy token may be on demand or on schedule. An example of a scheduled movement of a true-copy token is seen in a bank with branches where local accounts are maintained in the daytime, but central postings from other banks take place at night. The situation regarding ownership, authorization of updating rights and the overall database administration and control becomes further complicated in the above types of environment.

3.4.2. Technological Factors Influencing Distributed Databases

Distributed databases exist in two types of networks: one where the sites are geographically separated and the others such as local area networks where the databases may reside on workstations, file servers, etc., within a short geographical span. The logical problems in either case are similar.

In order to plan appropriately for a logical integration of diverse applications and databases a substantial modeling effort is required. Models for all significant existing tasks must be established using actual documentation and verified using programmer and user knowledge as well as formal notions of consistency. Models must also be constructed for new intended applications and considered in conjunction.

The research on the "logical distribution of data" has recently resulted in models for horizontal and vertical fragmentation of data objects [20,80]. They are based on the following inputs:

(i) knowledge of the centralized schema
(ii) specification of important queries/transactions on the database
(iii) users' and designers' knowledge regarding good candidates for distribution.

A methodology for an optional distribution of data that minimizes the total cost (transmission + data access) of transaction processing has been developed. Heuristics have been proposed to introduce replication. Most of the above work has been implemented in the form of working program aids. The prospects for application of these models to realistic problems seem promising.

Operations integration will be more difficult. Commercially, hardly any system today manage distributed databases. Algorithms for management of replicated data are well understood [39]; however, they are almost nonexistent in current generalized systems. To share data among distinct users, a common solution adopted in industry is to build "bridges" explicitly. These bridges may be specific, often moving files created for that purpose between systems.

The design of distributed transactions is critical for good response and consistency. Known query types are decomposed into component transaction sections; each transaction section is placed with the database it needs; some sections invoke and correlate information from the other sections. Careful planning can minimize problems of interference [10]. If consistent models are available in stored form on the participating databases, then remote queries can be phrased in terms of these queries and in a truly distributed fashion.

3.4.2.1. The Validity of Distribution. If an organization has a geographically distributed operation, it does not necessarily require a distributed database system. A collection of decentralized databases where there is a periodic transfer and sharing of data between a central "host" database and local databases may be called for. For instance, consider the operation of a set of warehouses where much of the data needed for the operation of a warehouse is used only locally. Information about shipments does not need to flow much faster than the shipments do. A small percentage of queries result in out-of-stock conditions and may be rapidly distributed to nearby warehouses by establishing a connection when needed. Management information for periodic distribution is best summarized locally, using globally established procedures, and can be transmitted at low priority and cost to a corporate office.

Fragmentation and partitioning of databases will reduce some of the complexity now needed in database management systems designed to deal with extremely large files. In numerical terms, for operations on databases that need sorting in some form, the n log n performance bound is less for m distributed systems holding n entities each ($m(n \log n)$) than for a central system of equal size ($(mn)\log(mn)$). When systems become small, they may need to depend to a greater extent on remote facilities for maintenance, logging, and recovery. Issues of communication failure and processor failure need detailed analysis and are still in the research domain.

3.4.3. The Data Restructuring and Conversion Approach to database Communication

Normally, a distributed database system is viewed as the distribution of the data as well as the database management function across multiple, tightly coupled nodes in a communications network. All nodes understand a common schema and a homogeneous set of commands as well as their related protocols.

In the realistic future, however, distributed data processing will be required among loosely coupled nodes that will communicate asynchronously. Furthermore, different nodes may use different DBMS's. To operate effectively in such an environment, three important facilities are required: a) a data dictionary system, b) a general data translation facility, and c) a store and forward communications service.

IBM's data extraction, processing, and restructuring system (XPRS) [53] is one example of a general data translation facility. XPRS implements the DEFINE data definition language, and the CONVERT data restructuring language. Using data translation techniques, users can extract data from a source database and map it to a target database. More and more "migration software" packages are becoming available. For example, Applied Data Research has a Package called DL/1 transparency that allows DL/1 (IMS) programs to run directly against their DATACOM/DB databases. A store and forward facility transports not only the data, but also the data translation program that together provide a "data translation schema" of the data. The data dictionary plays the important role of relating the data translation schema to the database schema for the given source and target database management systems. In effect, the data translation and data dictionary facilities can serve as a "bridge" between heterogeneous DBMS's.

There are many scenarios for distributing the data mapping process. For example, files could be extracted from the source database, restructured in the source system, and then sent to the target system. Alternatively, if the data translation system resided in the target system, the unstructured source data could be sent (along with their definitions) to the target system for restructuring and insertion into the target database.

3.4.4. The Federated Approach to Database Communication

One of the most significant trends and important challenges of the next decade in the area of computerized database systems is an effective support of decentralized collections of data. Recent work on distributed databases has focused on techniques to store and access data at various nodes in a computer network, as if that data were part of a single logical database. These efforts concentrate on the problems of distributed query processing where queries must be appropriately modified, routed to different sites, partial results framed and collected of the originating site. The global dictionary keeps the critical information regarding physical distribution.

Logical decentralization is an equally important concern. In response to the need for providing a loosely coupled decentralized database system, a new database system (software) architecture has been developed, termed federated database systems. The goal of the federated database concept [47] is the logical decentralization and partial integration of databases.

A federated database consists of a number of logical components, each having its own user-level structural specification (component schema). The components of such a federation are related but independent, and they may not be disjoint. Typically, a component corresponds to a collection of data needed by a particular user or application, or a collection of closely related applications. The components in a federation are tied together by one or more unified or federal schemas that describe which parts of the local databases are to be included in a particular federation. The federal controller is a database system functional module that supports communication and translation of data among the components of a federation, based on the federal schema(s). At Honeywell Corporate Research Center, research is under way to develop the tools for schema mapping and integration [30,31].

An important use of the prototype federated database system development aid described above is in establishing controlled sharing among existing databases. To accomplish this, a semantic schema is defined for each component database, and the tool is then used to define a federal schema based on these component schemas. A prototype federal controller, an integral part of the development aid, will automatically support data communication and translation among the components, via the federal schema.

The last step in logically connecting the component databases is to develop a mapping from the actual database structure of each component to its corresponding semantic schema; this can either be done on a case-by-case basis; or mappings to generic classes of general-purpose database systems can be developed (e.g., to a relational database management system, to CODASYL DBTG, etc.).

The federation concept has a much wider applicability than merely a method of supporting logical database decentralization: the technique of federating information is a form of abstraction that can be used for structuring both data and programs. When used as an abstraction technique, federation facilitates the construction of complex collections of data from simple collections of data, and the construction of complex program modules from simple modules. Moreover, the techniques of federation abstraction can be applied to form hierarchies and other complexes of data and programs.

4. Conclusions

The essence of this paper came from the discussions of the Logical Database Design panel at the DBD III workshop in Ft. Lauderdale in October 1980. Since then, although there have been more research papers published in the area of semantic data models, requirements modeling and analysis, logical database design, physical database organization, distributed database design and allocation, as well as on data dictionaries, we believe that the basic problems and approaches we had discussed then are very much valid today. The conclusions reached by the panel back in 1980 do apply today and are stated at the outset in this paper. The material in the paper has been thoroughly revised since the panel put together a draft report.

The state of the art of data dictionaries today has not conceptually advanced that much over the last several years. One can see, however, that dictionaries have become an integral part of the Information Resource Management Function in most large organizations. In this paper we have tried to take a stock of the situation in the areas of requirements definition and analysis, information modeling, database design and administration, design tools, database integration and communication. In each of these areas we see data dictionary systems performing a vital role of coordination and integration. We have elaborated how this would happen by enhancing the meta-data management capabilities of the dictionary systems. While the technology is rapidly advancing in each of the above areas, a coordinated effort to harness those technologies can only become feasible with adequate support from data dictionaries. We hope that the dictionaries of the future will provide all the desirable functionalities and qualities that we outlined in this paper. A rapidly growing involvement of the data dictionaries will have a major impact on future information resource management.

Acknowledgement

The authors wish to thank all the panel members at the DBD III workshop for providing the basis for this paper. Thanks also go to Ed Sibley for his painstaking editorial work on earlier drafts of the paper. Finally, Sharon Wilson deserves thanks for her patience in preparing the manuscript.

References

[1] T.F. Aadstadt, G. Schylstad, and Solvberg, "CASCADE - A Computer-Based Documentation System," *Information System Analysis and Design*, Bubenko, Langefors, Solvberg, eds., Lund, Sweden, 1972.

[2] A. Albano, V. DeAntonellis, A. DiLeva, eds., *ComputerAided Database Design*, North Holland, 1985.

[3] M.M. Astrahan, et al., "System R: A Relational Approach to Database Management," *ACM Trans. on Database Systems* (*TODS*), 2: 2, 97–137 (June 1976).

[4] P. Atzeni, C. Batini, M. Lenzerini, and F. Villanelli, "INCOD: A System for Conceptual Design of Data and Transactions in the Entity-Relationship Model," *Proc. Second International Conference on Entity-Relationship Approach*, Washington D.C., (October 1981), 379–414.

[5] Auxtron Computer Enterprises, Inc. 2. *Auxco Project Management System: User Reference Manual, 1972.*

[6] J. Backus, "Can Programs Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs," *Communications of ACM*, 21: 8 (August 1978), 613–641.

[7] C. Batini, and M. Lenzerini, "A Methodology for Data Schema Integration in the Entity-Relationship Model", *IEEE Trans. on Software Engineering*, SE-10: 6, (November 1984).

[8] C. Batini, M. Lenzerini, and G. Santucci, "A ComputerAided Methodology for Conceptual Database Design," *Information Systems*, 7: 3, 1982, 265–280.

[9] E.W. Bakke, "The Human Resources Function", pp. 174–175, in E. Wight Bakke, Clark Kerr, and Charles W. Anrod, *Unions, Management and the Public*, 2nd edition, (Harcourt, Brace and Company, New York, 1960).

[10] P.A. Bernstein, D.W. Shipman, and J.B. Rothnie, Jr., "Concurrency Control in a System for Distributed Databases," *ACM Trans. on Database Systems*, 5: 1, (March 1980), 18–51.

[11] B. Breutman, E. Falkenberg, and R. Mauer, "CSL: A Language for Defining Conceptual Schemas," in *Data Base Architecture*, G. Bracchi and G.M. Nijssen, eds., (North-Holland, Amsterdam, 1979) 237–256.

[12] The British Computer Society, "Data Dictionary Systems Working Party Report," *Data Base and SIGMOD Record*, (1977).

[13] M.L. Brodie, "*Data Abstraction, Database and Conceptual Modeling: An Annotated Bibliography*," NBS Special Publication 500–59, Washington, D.C. (May 1980).

[14] R. Brown, R. Morrison, R. Ramey, F. Sullivan, and C. Patrick, *Architects Manual, ICAM Definition Method*, (IDEF Version 1), Aug. 1979.

[15] P. Buneman, and R.E. Frankel, "FQL - A Functional Query Language," *Proc. 1979 ACM-SIGMOD Conf.*, Boston, Mass., (May 1979), 52–57.

[16] P. Buneman, and A. Motro, "Constructing Superviews," *Proc. 1981 ACM-SIGMOD* Conf., Ann Arbor, MI, (May 1981), 56–64.

[17] J. Bubenko, and O. Kollhammer, "CADIS - Computer-Aided Design of Information Systems, in *Computer-Aided Information Systems Analysis and Design*, Bubenko, Langefors, Solvberg, eds., (Lund, Sweden, 1975), 119–140.

[18] A.F. Cardenas, and M.H. Pirahesh, "Data Base Communication in a Heterogeneous Data Base Management System Network," *Information Systems*, Vol. 5, 55–79 (1980).

[19] S. Ceri, ed., *Methodology and Tools for Database Design*, North Holland, 1983.

[20] S. Ceri, S.B. Navathe, G. Wiederhold, "Distribution Design of Logical Database Schemas," *IEEE Trans. on Software Eng.*, SE-9: 3, July 1983.

[21] D.D. Chamberlin, et. al., "SEQUEL 2: A Unified Approach to Date Definition, Manipulation, and Control," *IBM Journal of Res. and Dev.*, 20: 2 (Nov. 1976).

[22] P.P.S. Chen. "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Trans. on Database Systems*, 1: 1, (March 1976), 9–36.

[23] Childers, "Database Design: A Survey of Logical and Physical Design Techniques," *Database*, ACM SIGBDP Newsletter, 15: 1, March 1983.

[24] E.F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, 13: 6, (June 1970), 377-387.

[25] E.F. Codd, "Extending the Database Relational Model to Capture More Meaning," *ACM Trans. on Database Systems*, 4: 4, (December 1979), 397–434.

[26] "Proc. of the Conf. on Data Dictionary Systems," *Computer Bulletin*, 2: 18 (December 1978).

[27] Daniel Cougar, "Evaluation of Business System Analysis Techniques," *Computing Surveys*, Vol. 5, No. 3, (September 1973), 167–198.

[28] Cullinet Corporation, Integrated Database Management system (IDMS) Brochure, 1975.

[29] Database Design Incorporated, Data Design Product Description, Ann Arbor, MI 48104, 1983.

[30] R. ElMasri, J.A. Larson, S.B. Navathe, "A Comprehensive Methodology for Database Schema Integration," Honeywell Computer Science Center, Research Report, 1986.

[31] R. ElMasri, J.A. Larson, S.B. Navathe, T. Sashidhar, "Tools for View Integration," *Database Engineering*, 7: 4, December 1984, 28–33.

[32] R. ElMasri, S.B. Navathe, "Object Integration in Logical Database Design," *Proc. First Int. Conf. on Data Engineering*, IEEE, Los Angeles, April 1984, 426-433.

[33] R. El-Masri, and G. Wiederhold, "Data Model Integration Using the Structural Model," *Proc. 1979 ACM-SIGMOD Int. Conf.*, Boston, Mass., (May 1979), 192–202.

[34] E. Falkenberg, "Concepts for Modeling Information," in *Modeling in Database Management Systems*, G.M. Nijssen, ed., (North Holland, Amsterdam, 1976).

[35] Fry, J.P., and Teorey, T.J., "The Database Designer's Workbench," Users Manual, University of Michigan (1980).

[36] T.J. Gambino, and R. Gerritsen, "A Database Design Decision Support System," *Proc. 3rd Int. Very Large Database Conference*, Tokyo, Japan (October 1977).

[37] C.T. Gane, "Data Design and Structured System Analysis," in *Tutorial and Software Design Techniques*, Third Edition, T. Freeman and A.I. Wasserman, eds., (IEEE Computer Society, 1980).

[38] C.T. Gane, and T. Sarson, *Structured Systems Analysis: Tools and Techniques*, Prentice Hall, 1979.

[39] H. Garcia-Molina, "Distributed Database Coupling," *Proc. Third USA-JAPAN Computer Conf.*, San Francisco, (September 1978), 75–79.

[40] Gen. Acctng. Office, "Comptroller General of the U.S.: Data Base Management Systems--Without Careful Planning There can be Problems," Washington, D.C. FGMSD-79-35 (June 1979).

[41] R. Gerritsen, "Tools for the Automation of Database Design," *Proc. NYU Symposium on Database Design*, (May 1978), 91–97.

[42] R. Gerritsen, "SEED Reference Manual," International Data Base Systems, Inc., Philadelphia (1978).

[43] R. Gerritsen, T.J. Gambino, and F. Germano, Jr., "Cost Effective Database Design: An Integrated Model," Working Paper 77-12-03, Dept of Decision Sciences, Wharton, U. of Penn. (December 1977).

[44] R.H. Godlove, "Impacting Management: Future Directions of DB and DP," *Infosystems*, June 1979, p. 68.

[45] P. Griffiths Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lories, T.G. Price, "Access Path Selection in a Relational Database Management System," *Proc. 1979 ACM-SIGMOD Conf.*, Boston, Mass., (May 1979), 23–24.

[46] M. Hammer, and D. McLeod, "Database Description with SDM: A Semantic Database Model," *ACM Trans. on Database Systems*, 6: 3, 351–386 (September 1981).

[47] M. Hammer, and D. McLeod, "On the Architecture of Database Management Systems," *Infotech State of the Art Report on Data Design* (1980).

[48] W. Hartman, H. Matthes, and A. Troem, *Management Information systems Handbook - ARDI*, McGraw - Hill, 1968.

[49] M.S. Hecht, "Metadata Design for an Extended Relational Model of Data," unpublished manuscript, Bell Laboratories, Holmdel, NJ (1980).

[50] M.S. Hecht, and L. Kerschberg, "Update Semantics for the Functional Data Model," Database Research Report #4, Bell Laboratories, Holmdel, NJ (1980).

[51] B.C. Housel, V. Waddle, and S.B. Yao, "The Functional Dependency Model for Logical Database Design," *Proc. 5th Very Large Database Conf.*, Rio de Janeiro, Brazil, (October 1979), 194–208.

[52] IBM, *IMS/VS publications*, White Plains, NY (1978).

[53] IBM, *Data Extraction, Processing, and Restructuring System*, Manual No. SH20-2178 (1979).

[54] International Computers Ltd., "CADES - Computer-Aided Design and Evaluation Systems," General Information Manual (UK Software and Literature Distribution Center, ICL, ICL House, Putney, London SW15 ISW).

[55] K.B. Irani, S. Purkayastha, and T.J. Teorey, "A Designer for DBMS-Processable Logical Database Structures," *Proc. Fifth Int. Very Large Database Conf.*, Rio de Janeiro, Brazil, (October 1979), 219–231.

[56] ISDOS Project, University of Michigan, *Problem Statement Language, Language Reference Manual*, PSA Version 4.2, May 1977.

[57] ISDOS Project, University of Michigan, *Problem Statement Analyzer, User Manual*, Version 4.2, May 1977.

[58] M.N. Jones, "HIPO for Developing Specifications," *Datamation*, March 1976, p. 112–125.

[59] R.H. Katz, "Database Design and Translation for Multiple Data Models," Memorandum No. UCB/ERL M80/24, University of California at Berkeley (June 1980).

[60] R.H. Katz, and E. Wong, "An Access Path Model for Physical Database Design," *Proc. 1980 ACM-SIGMOD Conf.*, Santa Monica, Calif., (May 1980), 22–29.

[61] H. Katzman, Jr., *System Design and Documehtation: An Introduction to the HIPO Method*, Van Nostrand and Reinhold Pub. Co., 1976.

[62] L. Kerschberg, A. Klug, and D. Tsichritzis, "A Taxonomy of Data Models," in *Systems for Large Data Bases*, P.C. Lockemann, and E.J. Neuhold, eds., (North Holland, Amsterdam, 1977).

[63] L. Kerschberg, D.A. Marchand, and A. Sen, "Information System Integration: Metadata Management Approach," *Proc. 4th Int. Conf. on Information Systems*, Houston, TX, (Dec. 1983).

[64] L. Kerschberg, E.A. Ozkarahan, and J.E.S. Pacheco, "A Synthetic English Query Language for a Relational Associative Processor," *Proc. 2nd Intl. Conf. on Software Engineering*, San Francisco, Calif., (October 1976), 505–519.

[65] L. Kerschberg, and J.E.S. Pacheco, "A Functional Data Base Model," Technical Report, Pontificia University Catolica do Rio de Janeiro, Rio de Janeiro, Brazil (February 1976).

[66] Y.E. Lien, "On the Semantics of the Entity-Relationship Model," *Proc. Int. Conf. on the E-R Approach to Systems Analysis and Design*, Los Angeles (December 1979).

[67] V.Y. Lum, et al., "1978 New Orleans Data Base Design Workshop Report," *Proc. Fifth Int. Conf. on Very Large Databases*, Rio de Janeiro, Brazil, (October 1979), 328–339.

[68] D. McLeod, "A Database Transaction Specification Methodology for End-Users," Technical Report 79-6, Computer Science Department, University of Southern California, Los Angeles, CA (December 1979).

[69] D. McLeod, "On Conceptual Database Modelling," in *Proc. of Workshop on Data Abstraction, Databases, and Conceptual Modelling*, M.L. Brodie, ed., Pringree Park, CO (June 1980).

[70] D. McLeod, and D. Heimbigner, "A Federated Architecture for Database Systems," *Proc. NCC*, AFIPS, 49, Anaheim, California, (1980) 283–289.

[71] D. McLeod, and R. King, "Applying a Semantic Database Model," *Proc. Int. Conf. on the E-R Approach to Systems Analysis and Design*, Los Angeles, (December 1979).

[72] D. McLeod, and R. King, "Semantic Database Models," in *Principles of Database Design*, S.B. Yao, ed., Prentice Hall (1985).

[73] D. McLeod, and J.M. Smith, "Abstraction in Databases," *Proc. of Workshop on Data Abstraction, Databases, and Conceptual Modelling*, M.L. Brodie, ed., Pringree Park, CO (June 1980).

[74] T. Minoura, "Resilient Extended True-Copy Token Algorithm for Distributed Database Systems," Ph.D. thesis, Stanford University (1980).

[75] M.F. Mitoma, and K.B. Irani, "Automatic Data Base Schema Design and Optimization," *Proc. First Very Large Data Base conf.*, Framingham, Mass. (September 1975).

[76] MRI Systems Corp., *System 2000 Reference Manual*, Austin, TX (1978).

[77] J. Mylopoulos, P.A. Bernstein, and H.K.T. Wong, "A Language Facility for Designing Database-Intensive Applications," *ACM Trans. on Database Systems*, 5; 2, 185–207 (June 1980).

[78] S.B. Navathe, "A Methodology for Generalized Database Restructuring," Ph.D. Dissertation, University of Michigan, 1976. (Available from University Microfilms, Ann Arbor, Mich., Order No. TSZ 7627, 577).

[79] S.B. Navathe, "Schema Analysis for Database Restructuring," *ACM Trans. on Database Systems*, 5: 2, (June 1980), 157–184.

[80] S.B. Navathe, S. Ceri, G. Wiederhold, J. Dou, "Vertical Partitioning Algorithms for Database Design," *ACM Trans. on Database Systems*, 9: 4, (December 1984)

[81] S.B. Navathe, R. ElMasri and J.A. Larson, "Integrating User Views in database Design," *IEEE Computer*, 19: 1, January, 1986.

[82] S.B. Navathe, and S. Gadgil, "A Methodology for View Integration in Logical Database Design," *Proc. 8th Very Large Database Conf.*, Mexico City, Morgan Kaufmann publishing (September 1982).

[83] S.B. Navathe, and J. Lemke, "On the Implementation of a Conceptual Schema Model within a Three Level DBMS Architecture," *Proc. NCC*, AFIPS, 48, New York, (1979), 697–708.

[84] S.B. Navathe, T. Sashidhar, R. ElMasri, "Relationship Merging in Schema Integration," *Proc. 10th Very Large Database Conf.*, Singapore, Morgan Kaufmann Publishing, (September 1984).

[85] S.B. Navathe, and M. Schkolnick, "View Representation in Logical Database Design," *Proc. 1978 ACM-SIGMOD Conf.*, Austin, Texas, (May 1978), 144–156.

[86] R.L. Nolan, *Managing the Data Resource Function*, West Publishing Co., 1974, pp. 3–4.

[87] D. Reiner et al., "The Database Design and Evaluation Workbench (DDEW)," *Database Engineering*, 7: 4, December 1984, 10–15.

[88] D.T. Ross, "Structured Analysis (SA): A Language for Communicating Ideas," *IEEE Trans. on Software Engineering*, SE-3: 1, (Jan. 1977).

[89] D.T. Ross, and J.W. Brackett, "An Approach to Structured Analysis," *Computer Decisions*, Sept. 1976, pp. 40–44.

[90] D.T. Ross, and K.E. Shooman, Jr., "Structured Analysis for Requirements Definition," *IEEE Trans. on Software Engineering*, SE-3: 1, (Jan. 1977).

[91] P. Scheurermann, G. Schiffner, and H. Weber, "Abstraction Capabilities and Invariant Properties Modelling within the Entity-Relationship approach," *Proc. Int. Conf. on the E-R Approach to Systems Analysis and Design*, Los Angeles, (December 1979).

[92] M. Schkolnick, and P. Tiberio, "Considerations in Developing a Design Tool for a Relational DBMS," *Proc. COMPSAC Conf.*, Chicago, (November 1979).

[93] L.S. Schneider, "A Relational View of the Data Independent Accessing Model," *Proc. 1976 ACM-SIGMOD Conf.*, Washington, D.C., (June 1976), 75–90.

[94] M.E. Senko, "Specification of Stored Data Structures and Desired Output Results in DIAM-II with FORAL," *1st Int. Very Large Database Conf.*, Framingham, Massachusetts, (1975).

[95] M.E. Senko, E.B. Altman, M.M. Astrahan, and P.L. Gehder, "Data Structures and Accessing in Data-Base Systems," *IBM Systems Journal*, 1, (1973), 30–93.

[96] D. Shipman, "The Functional Data Model and the Data Language DAPLEX," *ACM Trans. on Database Systems*, 6: 1, (March 1981), 140–173.

[97] N.C. Shu, B.C. Housel, R.W. Taylor, S.P. Ghosh, and V.Y. Lum, "EXPRESS: A Data Extraction Processing, and Restructuring System," *ACM Trans. on Database Systems*, 2: 2, (June 1977), 134–174.

[98] E.H. Sibley, and L. Kerschberg, "Data Architecture and Data Model Considerations," *Proc. NCC*, AFIPS, 46, Dallas, Texas, (1977), 85–96.

[99] J.M. Smith, P.A. Bernstein, Y. Dayal, N. Goodman, T. Landers, K.W.T. Lin, and E. Wong, "MULTIBASE - Integrating Heterogeneous Distributed Database Systems," *Proc. NCC*, AFIPS, 50, Chicago, (1981), 487–499.

[100] J.M. Smith, and D.C.P. Smith, "Database Abstractions: Aggregation and Generalization," *ACM Trans. on Database Systems*, 2: 2, (June 1977), 105–133.

[101] SofTech, Ic., *An Introduction to SADT - Structured Analysis and Design Techniques*, Waltham, Mass., Feb. 1976.

[102] M. Stonebraker, and K. Keller, "Embedding Knowledge and Hypothetical Data Bases into a Data Base System," *Proc. 1980 ACM-SIGMOD Conf.*, Santa Monica, Calif., (May 1980), 58–66.

[103] M. Stonebraker, E. Wong, P. Kreps, and G. Held, "The Design and Implementation of INGRES," *ACM Trans. on Database Systems*, 1: 3, (1976), 189–222.

[104] S.Y.W. Su, "SAM*; A Semantic Association Model for Corporate and Scientific-Statistical Databases," *Information Sciences* 29, (1983), 151–199.

[105] S.Y.W. Su, and H. Lam, "Transformation of Data Traversals and Operations in Application Programs to Account for Semantic Changes of Databases," *ACM Trans. on Database Systems*, 6: 2, (June 1981), 255–294.

[106] S.Y.W. Su, and H.D. Lo, "A Semantic Association Model for Conceptual Database Design," *Proc. Int. Conf. on the E-R Approach to Systems Analysis and Design*, Los Angeles, (December 1979), 147–171.

[107] D. Teichroew, E.J. Rataj, and E.A. Hershey, III, "An Introduction to Computer-Aided Documentation of User Requirements for Computer-Based Information Processing Systems," ISDOS Working Paper No. 72, March 1973.

[108] T.J. Teorey, and J.P. Fry, "The Logical Record Access Approach to Database Design," *ACM Computing Surveys*, 12: 2, (June 1980), 179–211.

[109] D.C. Tsichritzis, "OFS: An Integrated Form Management System," *Proc. 6th Very Large Database Conf.*, Montreal, Canada, (October 1980), 161–166.

[110] G. Wiederhold, "Management of Semantic Information for Databases," *Proc. Third USA-Japan Computer Conf.*, San Francisco, (September 1978), 192–197.

[111] G. Wiederhold, *Database Design*, 2nd Edition, McGraw-Hill, New York (1983).

[112] S.B. Yao, S.B. Navathe, and J.L. Weldon, "An Integrated Approach to Database Design," in *Database Design Techniques: Requirements and Logical Structures*, in Lecture Notes in Computer Science, #132, (Yao, Navathe, Weldon, Kunii, eds.), Springer-Verlag, New York, 1982.

[113] S.B. Yao, V. Waddle, and C. Housel, "View Modeling and Integrating Using the Functional Data Model," *IEEE Trans. on Software Engineering*, SE-8: 6, (November 1982), 544–553.