Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 1987 Proceedings

International Conference on Information Systems (ICIS)

1987

A FRAMEWORK FOR DEDUCTIVE DATABASE DESIGN IM DECISION SUPPORT SYSTEMS

Joobin Choobineh
Texas A&M University

Arun Sen
Texas A&M University

Follow this and additional works at: http://aisel.aisnet.org/icis1987

Recommended Citation
Choobineh, Joobin and Sen, Arun, "A FRAMEWORK FOR DEDUCTIVE DATABASE DESIGN IM DECISION SUPPORT SYSTEMS" (1987). *ICIS 1987 Proceedings*. 44.
http://aisel.aisnet.org/icis1987/44

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 1987 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

A FRAMEWORK FOR DEDUCTIVE DATABASE DESIGN IN DECISION SUPPORT SYSTEMS

Joobin Choobineh and Arun Sen
Department of Business Analysis and Research
College of Business Administration
Texas A&M University

ABSTRACT

A three-level framework for design and implementation of deductive database management systems is described. The three levels consist of the abstraction, for abstracting the real world semantics, the language, for man-machine communication, and the environment, for specifying the hardware/software environment. This framework is applied to some representative systems. Based on the results, an architecture for a deductive database management system is proposed.

1. INTRODUCTION

The term "decision support" first began to appear in titles of research papers and in conferences in the early 1970s (Bonczek, Holsapple and Whinston 1981). Computers were usually integrated into such systems as a support mechanism; as a whole, these came to be known as Decision Support Systems (DSSs).

A DSS is defined as a collection of several tools: data management, analytical techniques, report writers, and visual displays (Sen 1983). Moreover, these tools need to communicate with one another so that they can collectively support the managerial decision-making process.

The traditional definition and the design considerations of a DSS have gone through a metamorphosis over the past decade, particularly because of users' insistence to make the software more and more "user-friendly." This trend can be seen in works by Donovan (1975), Elam (1979), Bonczek, Holsapple and Whinston (1981), Konsynski (1980), Lee (1985), and others. The trend is more toward building a smart DSS, using the notions of artificial intelligence. As DSS needs a database system, this new impetus in DSS has forced researchers to look into intelligent database systems.

Intelligent database research can be decomposed into three broad categories: (i) deductive database management system, (ii) smart database interface, and (iii) database system enhancement. Deductive Database Management Systems (D-DBMS) are database systems that can also perform deductive operations. These operations are needed for problems that cannot be solved by the traditional database systems. These problems include null value representation (Zaniolo 1981), incomplete data representation (Levesque 1981), representation of complex objects (Zaniolo 1981), virtual data type manipulation (Chang 1981), and heuristics representation (Kellogg 1984, 1986). Smart interfaces include systems that are intelligent and can be interfaced with a traditional database system. For example, a natural language processor acting as a front-end to a database system can act as a smart interface. Database enhancement includes areas such as query optimization (Aho and Ullamn 1979) and incremental query formulation (Codd 1978).

The objective of this paper is to study the deductive database systems for decision support. The connection between database and decision support systems has been thoroughly established (Carlson 1977; Donovan 1976; Sen 1983). However, the user-friendliness issue of the DSS is forcing the database research to include "something more" than just data retrieval and update. Some of these extra things are discussed above in connection with the deductive database.

Section 2 describes a framework for the design of D-DBMS. In Section 3, some representative systems are surveyed and presented in a tabular form according to the framework which is developed in Section 2. Section 4 introduces our goals and an architecture for a D-DBMS.

2. A FRAMEWORK FOR DESIGN

In this section, we develop a framework to design a D-DBMS. The framework will first be used to classify various D-DBMS which have been proposed or implemented. Later it will be applied to describe the architecture of DBFLEX. There are three levels in our framework. They are the abstraction level, the language level, and the environment level. The abstraction level includes relevant components to represent the real world. The language level is the user interface to the environment. The environment level depicts the residency of the evaluative and deductive components of a D-DBMS. These three levels roughly correspond to the three levels of knowledge system, language system, and problem processing of Bonczek, Holsapple and Whinston (1981).

2.1. The Abstraction Level

To study the abstraction level of D-DBMS, we first determine the components of the abstraction. In Table 1, the x-axis has the support components, while the y-axis is the problem type. If the D-DBMS is to be designed for decision support, one needs to know the type of support it can provide for various types of decision related problems. These problems can be data oriented, or may need process descriptions and enterprise level information. We think that all organizations will have these three levels of problem complexity. For example, at data level of the inventory management problems, one can ask "how much stock do we have for part number 562?" At process level, one finds that the scope is somewhat broadened. The query at this stage will be "how much should we order next month?" This query recognizes the fact that the ordering process is connected with the inventory management process. At the enterprise level, one finds queries like "should Mr. Jones be allowed to initiate this purchase order?"

To solve these various problems, one needs support tools. These support components are listed in the x-axis. We start with the *fact* component. This involves only the data types that are of interest to us. The next level is *general* laws. We define general laws as real world rules which should be followed. These laws are typically common knowledge and are available to everybody. General laws include constraints as well as deductive rules. The final level is expert's knowledge. This is exclusive to those individuals who are experts and is not a common knowledge. In the AI community, it is also known as *heuristics*. Pearl (1984, page vii) defines heuristics as "...stand[ing] for strategies using readily accessible though loosely applicable information to control [the] problem-solving process in human beings and machine[s]." Heuristics are rules of thumb which aid a problem solver in finding satisfiable solutions to the problems, or to reduce the search space so that a solution can be found faster.

The entries in each cell define the abstraction level components of the D-DBMS. Typically, the traditional database systems cover the cells (1,A) and (1,B) at the data level. Semantic data model research (Abrial 1974; Brodie, Mylopoulos and Schnidt 1984; Tsichritzis and Lochovsky 1982) has pushed us to include cells (2,A) and (3,A). We follow Widerhold (1984) to capture the components for the rest of the grid.

We now define these components of each cell in Table 1.

Table 1. The Abstraction Level of Deductive DBMS

PROBLEM TYPE \ SUPPORT COMPONENTS		fact A	general laws B	expert's knowledge C
data level	1	.object types .hierarchies of object types	.domain .structural .operations	not appl.
process level	2	.event graph .form flow	.procedural .application -specific .derived	.procedural .application -specific .derived
enterprise level	3	.object-graph	.enterprise .derived	.enterprise .derived

Definition 1. *Object type* (1,A) is an abstraction of the real-world. It is a Cartesian product of uniquely named attributes, not all necessarily of the same domain. For example, EMPLOYEE is an object type. Its attributes are employee number, name, address, telephone number, salary, etc. The object types can be kept in hierarchies (Smith and Smith 1977).

Definition 2. *Domain Laws* (1,B) are a set of rules that maintains the domain integrity of the object types. Examples of domains are integers, reals, time, etc.

Definition 3. *Structural Laws* (1,B) are defined to be the knowledge we have about dependencies and constraints among the data, restricting ourselves to general and intentional information. An example of a structural knowledge concept is a functional dependency such as employee --> department.

Definition 4. *Operations Laws* (1,B) are the rule set that maintains data integrity upon traditional database operations, such as insert, delete, retrieve, and modify.

Definition 5. *Event Graph* (2,A) is a graph that shows the natural progression of events in a process. A *Form Flow* (2,A) diagram has also been used along with the event graph to abstract process information in business oriented problems (Sen and Kerschberg, forthcoming; Tsichritzis 1982).

Definition 6. *Object Graph* (3,A) is a graph of object types that is used to capture the data types at the enterprise level. This has been used by De and Sen (forthcoming) to capture internal control semantics.

The next several definitions do not distinguish between the general laws and expert's knowledge as they can be used in both.

Definition 7. *Procedural Knowledge* (2,B) and (2,C) is the knowledge about appropriate methods and procedures, given some set of data. Many decisions must be made to select and properly invoke the computational procedures which will produce the desired result for a query. Making the wrong choice can lead to processing failures, errors in the result, and wasted resources.

Definition 8. *Application-specific Knowledge* (2,B) and (2,C) is potentially a large body of knowledge that is associated with each application. This is potentially unbounded for an application.

Definition 9. *Derived Knowledge* (2,B), (2,C), (3,B) and (3,C) is the knowledge that is not explicitly stored in the database and is deduced from the explicitly stored information using different rules.

Definition 10. *Enterprise Knowledge* (3,B) and (3,C) is the knowledge at the highest level of abstraction. It involves rules that are typically used by the strategic managers.

There are two other components that are not explicit in Table 1. They are control structure and deduction technique. We define the *control structure* as a mechanism that actually guides the search process that is inherent in any AI-oriented system. Various kinds of control include forward chaining (starting from the start states), backward chaining (starting from the goal states), bi-directional (forward and backward chaining) and opportunistic (special case of bi-directional). Further elaboration on these techniques can be found in any introductory AI textbook.

Deduction technique defines the method in which the deduction is to be carried out. For example, in predicate logic, one typically uses the resolution principle. In rule-based systems, researchers have used improved search techniques coupled with good pattern-matching. For inexact reasoning, certainty factor and Dempster-Shafer techniques have been used (Shortliffe 1976).

Notice how similar Table 1 is with Gorry and Scott Morton's (1971) framework. Support components are categorized in a hierarchy following Anthony's (1965) classification, with fact and some general laws suitable for operations people. Expert's knowledge follow strategic planners. In the problem type axis, we have exploited Gorry and Scott Morton's classification: data level corresponds to the structured world, and enterprise level corresponds to the unstructured world.

2.2 The Language Level

Various types of languages have been proposed to interface to a deductive database. They range from traditional procedural ones like Pascal, to specification oriented languages like Prolog, and to object oriented languages such as SmallTalk.

Procedural languages do not have deductive capabilities per se. They can be used to arithmetically derive new facts from old, and to reason from premises to goals by if-then-else type statements. They do not, however, have any data modeling capabilities and/or built-in database management system. A good programmer can code any of the deductive capabilities of other languages in a conventional programming language, but that is similar to rewriting the interpreter of deductive languages in the source procedural language.

The most popular **logic based language** is Prolog (Clocksin and Mellish (1981). Prolog is based on horn clauses. A horn clause is of the general form: "IF A & B & C & ... THEN G," in which there are no free and no existentially quantified variables. That is, all the variables are universally quantified. A horn clause is limited to at most one conclusion and must have one or more premises. In Prolog notation, the above statement is represented as: "G :- A, B, C, ..."

Lisp based languages for deductions are very popular in artificial intelligence but they have not been as popular for deductive data modeling and language development. Their main thrust has been in the development of natural language interfaces to databases. If deducing user intentions can be considered deduction then the lisp based languages can be included in the language category of our framework.

Frame based languages have gained more popularity recently due to their rich representation mechanism and semantics (Fikes and Kehler 1985). A frame system (Minsky 1975) is a data structure organized as a semantic network in which each node is a frame which represents an object's definitions as well as its behavior. The semantic network is normally organized as a generalization hierarchy. Examples of frame based languages are FRL (Goldstein and Roberts 1977), KRL (Bobrow and Winograd 1977), KEE (Kehler and Clemenson 1984), and UNITS (Stefik 1979).

Object oriented languages are closely related to the frame based languages with the addition of icons and messages passing between objects (Stefik and Bobrow 1986). In this sense, an object oriented language may be considered a richer language processor and user interface than the frame based systems. However, from a modeling standpoint, they are not much stronger than the frame based systems.

Expert system shells, like EMYCIN, M1, OPS5, etc., are very popular in expert systems applications development. Various researchers have envisioned a possible connection between database and expert systems (Kerschberg 1984).

2.3. The Environment Level

Different types of environments can be used to implement the D-DBMS. Issues at this level include selection of hardware, software, and their coupling environments. A D-DBMS can use homogeneous or heterogeneous hardware. Homogenous hardware is a single processor, whereas heterogeneous hardware is a multi-processor.

The software environment can also be homogenous or heterogeneous. If the entire system is written in one language, it is homogenous. Otherwise, the environment is heterogeneous.

Vassiliou, Clifford and Jarke (1985) describe two types of coupling of the evaluative component (traditional DBMS features) and the deductive component. They are loose coupling and tight coupling. **Loose coupling** is used when a snapshot of data from an existing database which is managed by a DBMS is needed by the deductive component. Such a strategy presents several practical advantages. However, it is not suitable if the portion of the database to be extracted is not known in advance. **Tight coupling** is used when the deductive component needs to access the database at various points during its operation. In this case, an online communication channel between the two components is required.

3. APPLICATION OF THE FRAMEWORK

In this section, we apply the framework of the previous section to representative Deductive DBMS which are reported in the literature. Tables 2 through 4 show the abstraction, language, and environment levels of the surveyed systems, respectively. The first column of each table contains the system's names or their acronyms. The first column of Table 2 also includes the year of publication and the reference to it. We estimate that there is approximately a one to two year lag between the implementation and publication of the systems. The rest of the columns are the components of each of the categories of our framework.

The first two columns of Table 1 correspond directly to the first two columns of Table 2. The column "Expert's knowledge" of Table 1 is included in the "Heuristic" column of Table 2. Heuristics may include some system related knowledge which is problem specific.

Table 2. Components of the Abstraction Level

	FACTS	GENERAL LAWS	HEURISTICS	CONTROL	DEDUCTION TECHNIQUES
SBRM 84 Azmoodeh, Lavington, & Standring 1984	Semantic net as a Triple store <ent 1, rel, ent 2> classes, meta classes, instances	Domains as entity classes, clausal form	Clausal Form	Not Implemented	Inheritance resolution
DEDUCE2 78 Chang 1976, 1978	Relations	FOPC, plus Numerical Quantifiers	Clausal Form	Not reported	Rewriting rules, a variation of resolution
MRPPS 78 Minker	Semantic net; relations	FOPC	Clausal From	Not reported	SL-Resolution for clauses HUSH-Resolution for Horn clauses
HOLMES 84 Getta, Rybinski 1984	Objects & N-ray Relationships between them	FOPC Close to Natural Language	FOPC close to natural language	Not reported	Inheritance Resolution
LDL 86 Tsur, Zaniolo 1986	NF2 relations	Extended Horn Clauses	Extended Horn Clauses	Not reported	Extended Resolution
PROSQL 86 Chang, Walker 1984	Relations	Horn Clauses	Horn Clauses	Backward Chaining	Resolution
TAXIS 80 Mylopoulos, Bernstein & Wong 1975	Token, classes, Meta Classes, Properties, Frames	Pre-requisite, Results, DB Actions, Domain Constraints	Not reported	Not reported	Inheritance
RX 84 Blum 1981,1982 Widerhold 1984, 1986	Frames Categories	Domain Contraints, generalization	Statistical knowledge on rules, causal knowledge	Not reported	Inheritance
KM-1 86 Kellogg 1986, 1984	Relations	FOPC	FOPC	Forward chaining Backward chaining Mix	Resolution
ROSIE 81 Fain, Gorlin, Hayes-Roth & Rosenschein 1981	Relations <ent1,ent2,ent3>	Rules	Rules	Forward & Backward Chaining	Pattern-matching
STROBE 86 Laufue, Smith 1984	Frame	Integrity constraint, generalization	Allowed	Not reported	Inheritance
PROBE 86 Dayal, Smith 1986	DAPLEX environment	Allowed	Allowed	Not reported	Not reported
SRL 86 Fox 1983,1986	Frame	Allowed	Allowed	Constraint directed multi-level search	Inheritance Constraint-Directed

3.1 The Abstraction Level

Table 2 shows how each system abstracts the real world. Most of the systems use a relational model for abstracting the object types and relationships between them. This is more so for systems which are based on logic due to the basic underlying mathematical foundations of the relational model and logic. Some more recent systems represent the facts with richer abstractions such as frames or non-normal relations. For instance, MRPPS and SBRM use the semantic network; TAXIS, RX, STROBE, and SRL use frames; and LDL extends logic to handle sets and non-normalized relations. We believe the trend is toward richer representations and away from simple relational models.

Most systems represent general laws as situation-action rules which can easily be formalized in predicate logic. However, some are more restrictive and limit this representation to horn clauses. One reason for this limitation is that most of the logic-based systems are written in Prolog. Similarly, heuristics are represented in clausal form in most systems.

The control mechanism, for systems which use unification and pattern matching to deduce new facts, can be forward chaining, backward chaining, or a mix of the two. Most of the systems do not report the control mechanism that they employ. Only KM-1 reports it employs all three mechanisms. Unification and resolution is the main deductive technique used by most. Variations of the resolution is used depending on the extent of the logic formalism. For instance, LDL uses an extended resolution since it has extended logic to handle sets and non-normal form relations. Inheritance is the next popular method of deduction. Considering the fact that the inheritance can be represented in logic, its importance becomes subordinate to logic formalism. SRL is the exception to the rest since it employs constraint directed deduction.

3.2 The Language Level

Table 3 depicts the language basis of the systems. Out of the thirteen systems, eight use logic or some variation of it, four use Lisp, four use frames, one uses an object oriented approach, and two are shells. We will report not only the language in which the system is written, but also the notions which are supported by that system. For instance, if a system is written in Lisp but it is a frame based system, then both frame and Lisp are marked in the body of the table for it.

3.3 The Environment Level

Aside from SBRM and PROBE, all systems report some implementation. Most of them, however, do not fully report the implementation environment. Hence, some entries in Table 4 were guesses. All systems reside on a homogeneous hardware except KM-1. We envision the proliferation of multi-processor environments due to the decrease in hardware cost and increase in demand for computing power and inferential applications.

Similar to the hardware environment, most systems are implemented through a single software. If a system resides in a heterogeneous hardware environment, it is likely that its underlying software is also heterogeneous. This is the case for KM-1. PROSQL is an exception to this norm. It is imple-

Table 3. Language Level-Implementation Languages

	Procedural	Logic Based	Lisp Based	Frame Based	Object Oriented	ES Shell
SBRM		X				
DEDUCE 2		X				
MRPPS		X				
HOLMES		X				
LDL		X				
PROSQL		X				
TAXIS	PASCAL			X		
RX			X	X		
KM-1		X	X			
ROSIE						X
STROBE			X	X	X	
PROBE		X				
SRL			X	X		

Table 4. The Environment of Implemention

	HARDWARE		SOFTWARE		COUPLING		
	HOMO	HETERO	HOMO	HETERO	LOOSE	TIGHT	COMMENTS
SBRM	Not implemented						A Framework for a smart DB machine
DEDUCE 2		Does not explain. It is the target language for RENDEVOUYZ (16)					Uses theorem proving tech. to resolve virtual rel. against base relations
MRPP	Univac 1108		SIMPL			One Language	Experimental System
HOLMES	X		X			X	Under Development
LDL	X		X			X	Prototype; pure Horn clauses
PROSQL	X		Prolog, SQL			Proposed	Embeds SQL in PROLOG
TAXIS	X		X			X	Extends data oriented semantics such as IS_A to procedures
RX	X		Interlisp			X	Multiple KB Management
KM-1		Xerox 1100, Britton-Lee IDM-500		Lisp and A Data Mgr	X		Interfaces a LISP machine to a DB machine
ROSIE	X		X			X	A general purpose rule base programming environment
STROBE	Xerox		Interlisp D			X	Extends an object oriented language to include integrity constraint manager
PROBE	Not implemented						A framework to develop advanced DBMS
SRL	X		Lisp			X	A frame based language with "schema" as its primitive

mented in a homogeneous hardware environment as a Prolog system with embedde
d SQL.

Coupling the database and the deductive component is tight for all systems with a homogeneous software environment. The question of loose versus tight coupling arises when the software is heterogeneous. For instance, the coupling is loose in PROSQL but it is shown how a tight coupling can be implemented. In KM-1 the coupling is also loose due to down loading of the needed data from the global database to a local database.

4. DBFLEX: A DEDUCTIVE DATABASE MANAGEMENT SYSTEM

This section, describes the architecture of DBFLEX, a deductive database management system, which we are presently in the process of designing and implementing. The following criteria are established for the development of DBFLEX:

1. support of inferencing and truth maintenance;
2. efficient processing of recursive queries;
3. support for non-normalized relations;

4. support for plausible reasoning which implies that the facts in the database are no longer 100% true;
5. ability to explain why and how questions, that is, why the system is pursuing this information, and how it arrived at this state;
6. facilities to retrieve, insert, delete, and update facts, and rules.

4.1 The Abstraction in DBFLEX

Facts in DBFLEX are represented as non-normalized relations which can be organized into various abstraction hierarchies. The non-normalized relational structure is chosen due to its natural correspondence to most real world business applications which are hierarchical in nature. The objects, which are represented as non-simple tuples, are further augmented by rules of behavior, which we called general laws and heuristics in our abstraction taxonomy.

Two types of implications are supported in DBFLEX. The situation-action rules are used for their side effects in enforcing integrity constraints and implementing triggers and alerters. The general rules are used for deriving values for virtual fields from the explicitly stored data. General rules are similar to view definition of modern relational DBMS but they can also be used to define recursive relationships.

In answering queries, as in KM-1, the control mechanism of DBFLEX uses forward chaining (what if), backward chaining (find), and bi-directional (given-find) techniques. The deduction techniques are based on resolution and inheritance.

4.2 The Language of DBFLEX

Our preliminary implementation language of DBFLEX is C. The user interface will be a variation of SQL. This includes augmentation of SQL with new data definition and manipulation constructs in order to be able to accommodate the above criteria. SQL is chosen due to its popularity and recognition as a standard database language (ANSI 1986).

4.3 The Environment of DBFLEX

DBFLEX will be implemented on a single SUN 3/160 workstation. Therefore, the hardware environment is homogeneous. The software will also be homogeneous. There will be one language which is used for both deduction and evaluation. A tight coupling will be employed. The system will be tightly integrated; data and rules use each other as the need may arise.

4.4 Architecture of the DBFLEX

Figure 1 depicts the architecture of the DBFLEX. Users interact with the system through a common interface which includes Data Definition, Data Manipulation, Rule Definition and Rule Manipulation. A planning component manages the distribution of user requests through three engines. The evaluative engine can search through various knowledge components for retrieval of explicitly stored knowledge. The deductive engine performs inferential search to satisfy deductive queries which involve derived facts. Upon an insert, delete, or update of any of the knowledge bases, the truth maintenance engine searches knowledge components for violation of constraint rules, conflicts in stored knowledge, and derivability of the new request from the knowledge bases.

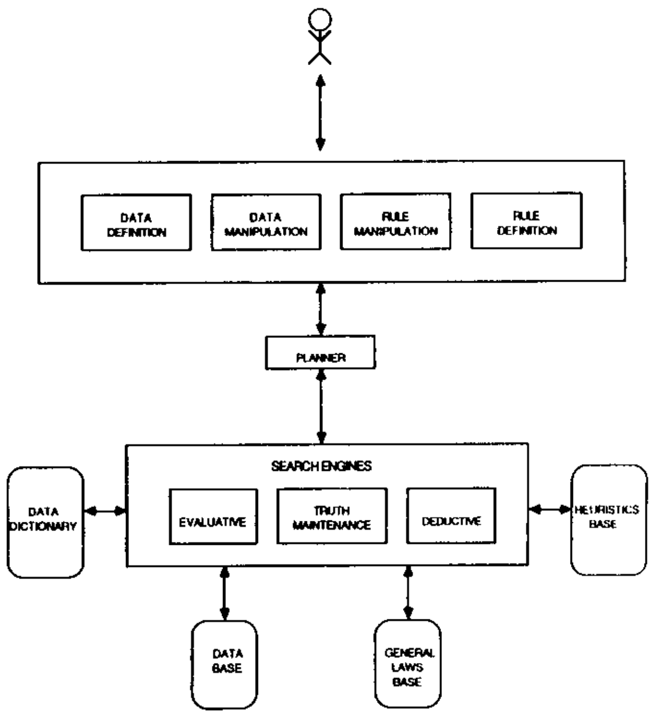

Figure 1. Architecture of the DBFLEX

The knowledge base of a system developed through DBFLEX will be composed of four components. The Data Dictionary contains descriptions of data such as their types, domains, derived or explicit, keys, and indexes. The Database contains the explicitly stored data. The General Laws Base contains the integrity constraint rules, triggers, and alerters as well as the deductive laws for inference of derived facts. The Heuristics Base contains the experts' knowledge about the procedures, application domain, and organization. This knowledge is typically judgmental.

5. CONCLUSION

We have presented a three level framework for the design and implementation of a Deductive Database Management System. Since a Deductive DBMS must support decision making, our framework was based and contrasted to the Gorry and Scott Morton framework. We tested this framework by reviewing the current deductive systems. The framework is used to develop an architecture for DBFLEX, a deductive relational database management system.

ACKNOWLEDGMENT

This research was partially supported by a grant from the Research Enhancement Funds in Knowledge and Manufacturing Systems at Texas A&M University.

REFERENCES

Abrial, J. R. "Data Semantics." In J. W. Klimbie and K. L. Koffman (eds.), *Data Base Management*, North Holland, Amsterdam, 1974, pp. 1-60.

Aho, A. V., and Ullamn, J. D. "Universality of Data Retrieval Languages." *Proceedings of the Sixth ACM Symposium on Principles of Programming Languages*, San Antonio, Texas, 1979, pp. 110-120.

American National Standards Institute. *Database Language SQL*. Document ANSI X3.135.1986.

Anthony, R. N. *Planning and Control Systems: A Framework for Analysis*. Harvard University Graduate School of Business Administration, Boston, 1965.

Azmoodeh, M.; Lavington, S. H.; and Standring, M. "The Semantic Binary Relationship Model of Information." In *Research and Development in Information Retrieval*, Proceedings of the Third Joint BCS and ACM Symposium, King's College, Cambridge, England, July 2-6, 1984, C. J. Van Rijsberg, Cambridge University Press.

Blum, R. L. "Displaying Clinical Data from a Time-Oriented Database." *Computers and Biomedical Research*, Vol. 11, No. 4, 1981, pp. 197-210.

Blum, R. L. "Discovery, Confirmation and Incorporation of Causal Relationships from a Large Time-Oriented Clinical Data Base: The RX Project." *Computers and Biomedical Research*, Vol. 15, 1982, pp. 164-187.

Bobrow, D. G., and Winograd, T. "An Overview of KRL, A Knowledge Representation Language." *Cognitive Science*, Vol. 1, No. 1, January 1977, pp. 3-46.

Bonczek, R. H.; Holsapple, C. W.; and Whinston, A. B. *Foundations of Decision Support Systems*. Academic Press, New York, 1981.

Brodie, M.; Mylopoulos, J.; and Schnidt, J. W. (eds). *On Conceptual Modelling*. Springer-Verlag, New York, 1984.

Carlson, E. D. "An Approach for Designing Decision Support Systems." *IBM Technical Report # RJ 1959 (27739)*, 1977.

Chang, C. L. "DEDUCE -- A Deductive Query Language for Relational Databases." In C. G. Chen (ed.), *Artificial Intelligence and Pattern Recognition*, Academic Press, New York, 1976, pp. 108-134.

Chang, C. L. "DEDUCE2: Further Investigations of Deductions in Relational Databases." In H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum Press, New York, 1978.

Chang, C. L. "On Evaluation of Queries Containing Derived Relations in a Relational Database." In H. Gallaire, J. Minker, and J. M. Nicolas (eds.), *Advances in Database Theory*, Plenum Press, New York, 1981, pp. 235-260.

Chang, C. L., and Walker, A. "PROSQL: A Prolog Programming Interface with SQL/DS." In L. Kerschberg (ed.), *Expert Database Systems--Proceedings of the First International Workshop*, Kiwah Island, South Carolina, 1984, pp. 233-246.

Clocksin, W. F., and Mellish, C. S. *Programming in Prolog*. Springer-Verlag, New York, 1981.

Codd, E. F. "Seven Steps to Rendevous with the Casual User." In J. W. Klimbie and K. L. Koffman (eds.), *Data Base Management*, North Holland, Amsterdam, 1974.

Codd, E. F. "How About Recently? (English dialog with relational databases using RENDEVOUS Version 1)" In *Databases: Improving Usability and Responsiveness*, Academic Press, New York, 1978, pp. 3-28.

Dayal, U., and Smith, J. M. "PROBE: A Knowledge-Oriented Database Management System." In M. L. Brodie and J. Mylopoulos (eds.), *On Knowledge Based Management Systems*, Springer-Verlag, New York, 1986.

De, P., and Sen, A. "Semantic Modeling for Internal Controls in Database Design." Submitted to *Journal of MIS*.

Donovan, J. J. "Data Base Systems Approach to Management Decision Support." *ACM Transactions on Database Systems*, Vol. 1, 1976, pp. 344-369.

Elam, J. J. *Model Management System: A Framework for Developmemt*. Technical Report #79-02-04, Department of Decision Sciences, The Wharton School, February 1979.

Fain, J.; Gorlin, D.; Hayes-Roth, F.; Rosenschein, S. J.; Sowizral, H.; and Waterman, D. *The Rosie Language Manual*. Technical Report #N-1647-ARPA, Rand Corp, Santa Monica, California, 1981.

Fikes, R., and Kehler, T. "The Role of Frame-Based Representation in Reasoning." *Communications of the ACM*, Vol. 28, No. 9, September 1975, pp. 904-920.

Fox, M. S. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Technical Report CMU-RI-TR-83-22, 1983.

Fox, M. S., and McDermott, J. "The Role of Databases in Knowledge-Based Systems." In M. L. Brodie and J. Mylopoulos (eds.), *On Knowledge Based Management Systems*, Springer-Verlag, New York, 1986.

Gallaire, H., and Minker, J. (Eds.). *Logic and Data Bases*. Plenum Press, New York, 1978.

Getta, J., and Rybinski, H. "HOLMES: A Deduction Augmented Database Management System." *Information Systems*, Vol. 9, No. 2, 1984, pp. 167-179.

Goldstein, I. P., and Roberts, R. B. "NUDGE: A Knowledge-Based Scheduling Program." *Proceedings of the Fifth International Joing Conference on Artificial Intelligence*, 1977, pp. 257-263.

Gorry, G. A., and Scott Morton, M. S. "A Framework for Management Information Systems." *Sloan Management Review*, Fall 1971, pp. 55-70.

Kehler, T. P., and Clemenson, G. D. "An Application Development System for Expert Systems." *Systems Software*, Vol. 3, No. 1, January 1984, pp. 212-224.

Kellogg, C. "The Transition from Data Management to Knowledge Management." *Proceedings of the International Conference on Data Engineering*, Los Angeles, April 24-27, 1984, pp. 467-472.

Kellogg, C. "From Data Management to Knowledge Management." *Computer*, January 1986, pp. 75-84.

Kerschberg, L. (Ed.). *Expert Database Systems--Proceedings of the First International Workshop*, Kiwah Island, South Carolina, 1984.

Klimbie, J. W., and Koffman, K. L. (Eds.). *Data Base Management*, North Holland, Amsterdam, 1974.

Konsynski, B. R. "On the Structure of a Generalized Model Management System." *Proceedings of the Eighteenth Annual Hawaii Conference on Systems Sciences*, 1980, pp. 630-638.

Laufue, G. M. E., and Smith, R. G. "Implementation of Semantic Integrity Manager with a Knowledge Representation System." In L. Kerschberg (ed.), *Expert Database Systems -- Proceedings of the First International Workshop*, Kiwah Island, South Carolina, 1984, pp. 333-350.

Lee, R. "Database Inferencing for Decision Support." *Decision Support System*, Vol. 1, 1985, pp. 57-68.

Levesque, H. J. "The Interaction with Incomplete Knowledge Bases: A Formal Treatment." *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, August 1981, pp. 240-156.

Minker, J. "An Experimental Relational Database System Based on Logic." In H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum Press, New York, 1978, pp. 107-147.

Minsky, M. "A Framework for Representing Knowledge." In P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

Mylopoulos, J.; Bernstein, P. A.; and Wong, H. K. T. "A Language Facility for Designing Database-Intensive Applications." *ACM Transactions on Database Systems*, Vol. 5, No. 2, June 1980, pp. 185-207.

Pearl, J. Heuristics, *Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley, Reading, MA, 1984.

Sen, A. "Decision Support Systems: An Activity-Oriented Approach." *Journal of Information Science*, Vol. 7, 1983, pp. 23-30.

Sen, A., and Kerschberg, L. "Enterprise Modeling for Database Specificaition and Design." *Data and Knowledge Engineering* (forthcoming).

Shortliffe, E. H. *Computer-Based Medical Consultation: MYCIN*. American Elsevier, New York, 1976.

Smith, J. M., and Smith, D. C. P. "Database Abstractions: Aggregation and Generalization," *ACM TODS*, Vol. 2, No. 2, June 1977, pp. 105-133.

Stefik, M. J. "An Examination of a Frame-Structured Representation System." *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, Kaufmann, Los Altos, CA, August 1979, pp. 845-852.

Stefik, M. J., and Bobrow, D. "Object-Oriented Programming: Themes and Variations." *The AI Magazine*, Vol. 6, No. 4, Winter 1986, pp. 40-62.

Tsichritzis, D. C. "Form Mnagement." *Communications of the ACM*, Vol. 25, No. 7, 1982, pp. 453-478.

Tsichritzis, D. C., and Lochovsky, F. H. *Data Models*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

Tsur, S., and Zaniolo, C. "LDL: A Logic-Based Data Lanaguage." *MCC Technical Report*, February 11, 1986.

Vassiliou, Y.; Clifford, J.; and Jarke, M. "Access to Specific Declarative Knowledge by Expert Systems: The Impact of Logic Programming." *Decision Support Systems*, Vol. 1, 1985, pp. 123-141.

Widerhold, G. "Knowledge and Database Management." *Computer*, January 1984, pp. 63-73.

Widerhold, G.; Blum, R. L.; and Walker, M. "An Integration of Knowledge and Data Representation." In M. L. Brodie and J. Mylopoulos (eds.), *On Knowledge Base Management Systems*, Springer-Verlag, New York, 1986.

Zaniolo, C. "Incomplete Database Information and Null Values: An Overview." *Proceedings of the Advanced Seminar on Theoretical Issues in Data Bases*, Cetraro, Italy, September 1981.