1987

Conceptual modeling for the designer/fabricator knowledge base /

Michele A. Krause
Lehigh University

Conceptual Modeling for the Designer/Fabricator Knowledge Base

by

Michele A. Krause

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1987

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 6, 1987
(date)

Donald J. Hillman
Professor in Charge

L. J. Vannucci
Chairman of Department

Acknowledgement

I would like to thank Professor D. Hillman for his guidance in the preparation of this thesis.

Table of Contents

Acknowledgement iii

Abstract 1

1. Introduction 3

1.1 A Structured Approach to Knowledge-Base Conceptual Modeling 3

2. Problem Definition 6

2.1 Introduction 6
2.2 Decision-Support Systems 7
2.3 System Domain 8

3. Inference Mechanisms 11

3.1 Introduction 11
3.2 Inference Engine 11
3.3 Inference Rules 12
3.3.1 Maintaining Appropriate Levels of Abstraction 12
3.3.2 Modeling of Inferential Knowledge 14

4. Database Models 16

4.1 Introduction 16
4.2 Nonrelational Database Models 16
4.2.1 Inverted List 17
4.2.2 Hierarchic 18
4.2.3 Network 18
4.3 The Relational Model 19
4.3.1 Structures 20
4.3.2 Integrity Constraints 21
4.3.3 Relational Algebra 22
4.4 Discussion 23

5. Semantic Modeling 26

5.1 Introduction 26
5.2 The Entity-Relationship Model 26
5.2.1 Semantic Concepts 26
5.2.2 Symbolic Objects 27
5.2.3 Data Manipulation Language 29
5.2.4 Integrity Constraints 29
5.2.5 Entity-Relationship Diagrams 31
5.3 The Extended Relational Model RM/T 32
5.3.1 Semantic Concepts 32
5.3.2 Symbolic Objects 33
5.3.3 Data Manipulation Language 34
5.3.4 Integrity Constraints 35

6. Conclusions 37

Bibliography 41

Vita 43

List of Figures

Figure 1-1:	Data Processing System	5
Figure 1-2:	Knowledge-Based System	5
Figure 4-1:	Relational Operators	25
Figure 5-1:	Regular Entity Relation EMPLOYEE	28
Figure 5-2:	Example Entity Relationship Diagram	32
Figure 6-1:	Model for Conceptual Analysis	40

Abstract

The overall objective of the research project of which this work is a component is to develop a real-time, decision-support system for the design and fabrication of large structural systems. This thesis surveys modeling tools which are appropriate for the development of the knowledge base conceptual model.

Of the database models discusssed in this work (relational, inverted list, network and hierarchic), the relational model is best suited for the designer/fabricator knowledge base because of its clearly-defined data manipulation operators and integrity rules, and tabular representation of data.

As a semantic extension to the relational model, the Extended Relational Model RM/T offers several advantages over the Entity-Relationship Model, including the use of system-defined surrogates and the availability of clearly-defined data manipulation operators and integrity rules. In addition, the model resolves ambiguities which result from distinguishing between entities and relationships as semantic concepts. RM/T does not provide an alternative to the diagrammatic technique offered by the Entity-Relationship Model; however, Entity-Relationship Diagrams can be easily adapted to represent the kernel, characteristic and associative entitites of RM/T.

To avoid focusing problems information in the knowledge base could be organized in a hierarchical fashion, where each node in the tree is a specialist which controls processing of rules within its domain. For example, the specialist tree might consist of the type of information normally contained in taxonomic rules, in which case conceptual modeling could be viewed as the process of "filling in" the specialist tree by modeling the transformation rules, state change rules and database facts which are associated with each specialist.

Further research is required to determine whether this approach is appropriate for the designer/fabricator knowledge base, and for decision-support systems in general.

Chapter 1
Introduction

1.1 A Structured Approach to Knowledge-Base Conceptual Modeling

The term "knowledge engineering" was first coined by Feigenbaum (1977) and refers to the process of reducing a large body of knowledge to a precise set of rules and facts. Unlike conventional programs which proceed according to a fixed algorithm and have no way of adapting to changing circumstances, knowledge-based systems apply information, acquired from human experts, in novel ways in different situations.

Figure 1-1, page 5, illustrates a typical data processing environment which consists of an application program connected to an I/O formatter and a database management system. Whereas data processing systems use conventional programs for computation, knowledge-based systems, such as the one depicted in Figure 1-2, page 5, separate computational steps from the control flow and put them in nonprocedural tables of rules. Such a system typically consists of a language handler which analyzes input and generates output, an inference engine that does deduction based on rules of logic rather than procedures, and a database management system which stores and retrieves data upon request.[1]

A second major difference between data processing and knowledge-based systems lies in the nature of the information stored in the database. In a logic-based representation, predicates reflect variety, whereas instances of predicates

---

[1]John F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine.* (Reading, Mass: Addison-Wesley Publishing Company, 1984), pp. 278-280.

refer to the population. For example, given the Prolog predicates:

```
customer(smith)
covered_by(smith,term_annuity)
```

"customer" and "covered by" reflect the variety of the database, whereas "Smith is a customer" and "Smith is covered by term annuity" refer to the population. In general, most knowledge-based system databases exhibit a large variety of facts with a more variable population, and therefore tend to be more "wide" and less "deep" than data processing databases.[2]

Because knowledge-based systems are complex and highly domain-specific, the conceptual modeling process typically takes place without the benefit of the structured approach typically used by system analysts. The lack of a generalizable approach to knowledge-base conceptual modeling suggests that the first step in the design process should be the selection of modeling tools which are appropriate to the system domain.

The remainder of this thesis explores this process within the framework of an ongoing research project at Lehigh University. Chapter 2 discusses the project domain. Chapter 3 discusses modeling of the system's inference mechanisms. Chapters 4 and 5 suggest tools and techniques relevant to the design of the database. Chapter 6 discusses the information presented and suggests possible extensions.

---

[2]Matthias Jarke and Yannis Vassilious, "Coupling Expert Systems with Database Management Systems" in *Artificial Intelligence Applications for Business: Proceedings of the NYU Symposium, May, 1983*, ed. Walter Reitman. (Norwood, New Jersey: Ablex Publishing Corporation, 1984), p. 70.

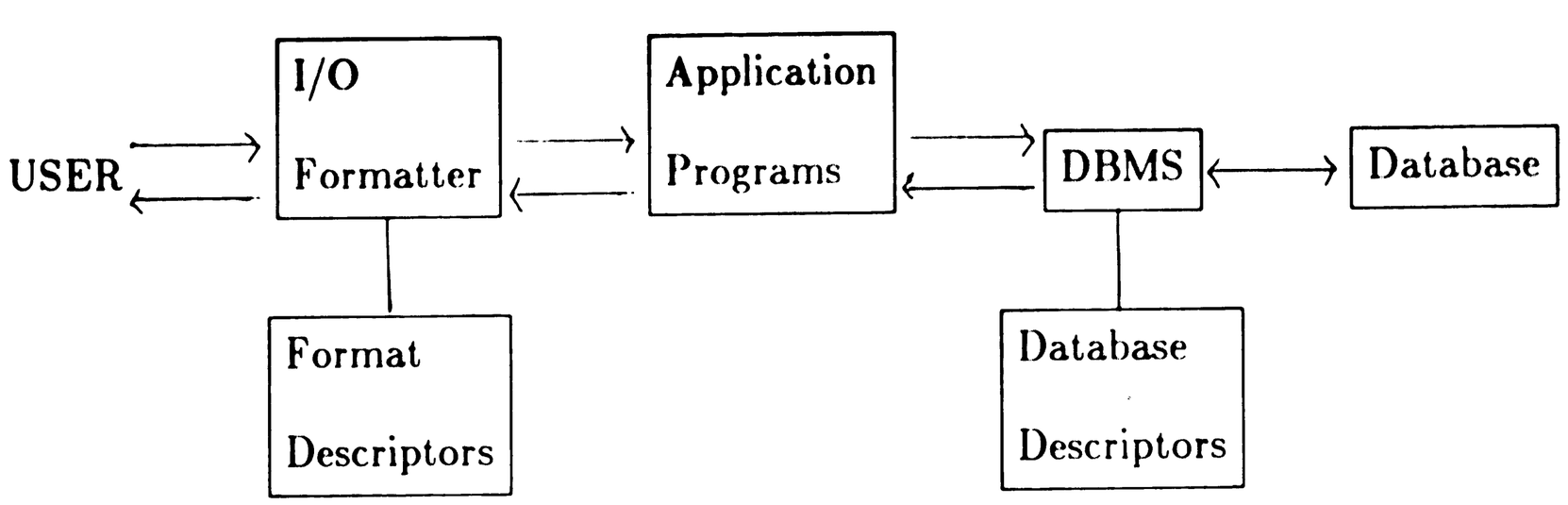

Figure 1-1: Data Processing System

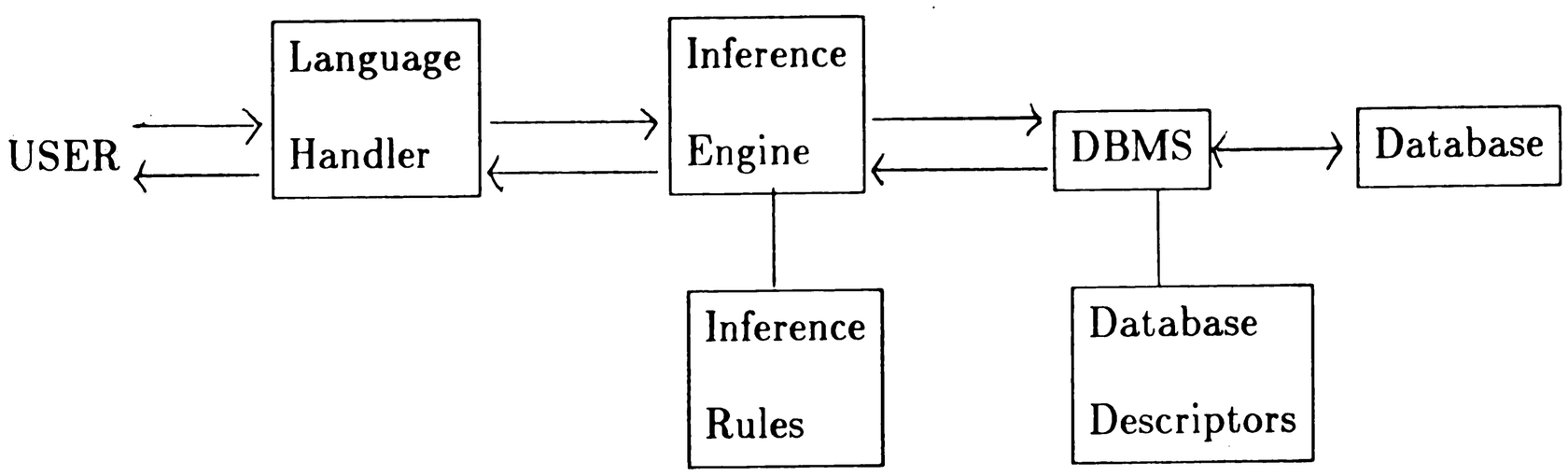

Figure 1-2: Knowledge-Based System

Chapter 2
Problem Definition

2.1 Introduction

The first step in the development of a conceptual model is to analyze the types of generic tasks which need to be accomplished by the system. The most frequently encountered generic tasks in knowledge-base design are:

- *CLASSIFICATION:* sort a large amount of data into categories, typically for diagnostic purposes.
- *DESIGN:* perform plan synthesis by searching for some combination of structures to fulfill a certain goal.
- *DECISION-SUPPORT:* aid decision making by exploring alternatives, making predictions and solving problems.[3]
- *PLANNING:* plan actions to achieve a goal.
- *INTERPRETATION:* analyze data to determine its meaning.
- *MONITORING:* analyze signals and plan appropriate actions.[4]

The overall objective of the research project of which this thesis is a component is to develop a real-time decision-support system for the design and fabrication of large structural systems. The purpose of this thesis is to suggest analysis techniques and modeling tools which are appropriate for the development of the knowledge-base conceptual model. The remainder of this chapter places this goal in context by providing general characteristics of decision-support systems (Section 2.2) and a more complete discussion of the system domain (Section 2.3).

---

[3]Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, pp. 280-281.

[4]Frederick Hayes-Roth, Donald A. Waterman and Douglas B. Lenat, eds., *Building Expert Systems.* (London: Addison-Wesley Publishing Company, Inc., 1983), p. 14.

2.2 Decision-Support Systems

Decision-support expert systems have their roots in sophisticated software packages designed to expand the capabilities of database query languages. *SAS* is representative of systems in this category, providing a programming language, statistical procedures, report generation facilities and line-printer graphics options, in addition to data management capabilities.

Whereas generalized statistical packages provide decision support through the analysis of past and present data, decision-support systems operate within a more specialized domain to make predictions of the future. Developed primarily for business applications, early decision-support systems use fixed algorithms, conventional programming techniques, and statistics to solve problems. Representative systems in this category include *BRANDAID*, which evaluates business decisions by relating strategies to sales and profits, and the *PORTFOLIO MANAGEMENT SYSTEM*, which evaluates decisions to buy or sell securities.[5]

The use of techniques from the field of artificial intelligence represents the most recent development in the evolution of decision-support systems. Systems in this category generally serve as intelligent front-ends to a knowledge base which contains expertise about a highly specialized domain. Predictions of the future are derived from the information in the knowledge base using surface or model-based reasoning, a variety of search techniques and domain-specific heuristics.

Rome, an expert system developed at Carnegie-Mellon University, is representative of systems in this category. Designed to circumvent the limitations of traditional financial-planning systems, *Rome*:

---

[5]Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, p. 284.

- differentiates good from bad consequences based on corporate goals.
- provides "intelligent" decision support for long-range planning through knowledge of the meaning of variables in financial models.
- presents not only the results of calculations, but a belief factor for input data and confidence factor for the results.[6]

The designer fabricator decision-support system most closely resembles this last category of systems in that it requires the development of a domain-specific knowledge base and the use of heuristics to process information. More specific information on the proposed system is presented in the next section.

2.3 System Domain

Designers and fabricators of large structural systems such as bridges and skyscrapers typically hold differing perspectives on a given construction task, a fact which leads to frequent mismatches between the designer's specifications and the fabricator's ability to build the components economically. The designer submits drawings and specifications to the fabricator, neither of which contain a great deal of information about the reasoning process which went into the design of the structure. Without this information the fabricator must second-guess the designer's intentions, a process which can lead to frequent and costly design changes.

The designer/fabricator decision-support system bridges the communication gap between designers and fabricators by providing them with more immediate access to one another's expertise. The system is scheduled to be developed as follows:

---

[6]Wendy B. Rauch-Hindin, *Artificial Intelligence in Business, Science, and Industry: Volume II - Applications*, (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1985), p. 35.

1. Develop a knowledge base which can support the differing perspectives of designers and fabricators.

    - Select construction experts
    - Identify design variables
    - Identify design constraints
    - Develop a conceptual model
    - Implement the conceptual model

2. Enhance the knowledge base through the addition of an interpreter which will allow designers and fabricators to communicate in one another's language.

3. Evaluate the system for performance and acceptability.

The system is currently in the first stage of development. Once selected, design and fabrication experts will identify design variables and constraints. Design variables are expected to include:

- Dimensions and characteristics of objects and their components
- Performance measures
- Cost effectiveness
- Functionality requirements
- Serviceability requirements
- Aesthetic requirements
- Long-term maintainability
- Environmental factors

Values of design variables are typically constrained in some way. Examples of design constraints include:

- Definitions (for example, dynamic loading)

- Models of the physical world (for example, stress-strain equations)
- Models of the performance of an object or its elements (for example, finite element modeling)[7]

After construction experts have been selected and design variables and design constraints have been identified, the conceptual model for the knowledge base can be developed and implemented. The next chapter begins the process of suggesting modeling tools which are appropriate to this task by discussing the proposed system's inference mechanisms.

[7]Donald Hillman, "Knowledge-Based Systems for the Designer/Fabricator Interpreter," Unpublished Manuscript, Lehigh University, Bethlehem, PA, 1986, pp. 1-6.

Chapter 3
Inference Mechanisms

3.1 Introduction

In order to fulfill its function as a decision-suppport system, the designer/fabricator knowledge base should be able to handle the following types of queries:

- What limits do the fabricator's assembly methods and available equipment place on the structural design?
- What design codes are in effect?
- Can a particular aspect of the design be changed? What would be the consequences of such a change?

After a brief discussion of the system's inference engine in Section 3.2, Section 3.3 discusses a framework for modeling and implementation of the inference rules in order to facilitate evaluation of these types of questions.

3.2 Inference Engine

The designer/fabricator knowledge base is scheduled to be implemented in Prolog, a programming language with a powerful built-in inference engine. Prolog (PRogramming language based on LOGic) is based upon a restriction of the first-order predicate calculus that permits only Horn clauses.[8] There are three basic statements in the language:

:- P.	Means P is a goal to be proved.
P.	Means P is an assertion.

[8] I. Futo, F. Darvas and P. Szeredi, "The Application of Prolog to the Development of QA and DBM Systems" in *Logic and Databases*, ed. Herve Gallair and Jack Minker. (New York: Plenum Press, 1978), p. 347.

P :- Q, R, S. Means Q and R and S imply P.

A Prolog program is a collection of clauses whose variables are considered to be universally quantified. Each clause has both a declarative and a procedural interpretation. For example, "P :- Q, R, S." can be interpreted declaratively as "Q and R and S imply P" or procedurally as "to satisfy P, first satisfy Q and R and S".

A given predicate is defined by one or more clauses and is represented by an AND/OR graph. Program execution involves a depth-first seach with back-tracking on these graphs, using the unification process based on the resolution principle [Robinson, 1965].[9]

The remainder of this chapter discusses the knowledge-base inference rules, with illustrative examples written in Prolog. For a more complete discussion of the syntax of Prolog see Clocksin and Mellish (1984).

3.3 Inference Rules

3.3.1 Maintaining Appropriate Levels of Abstraction

Prolog's rule-based architecture works well when relatively little complex coupling exists between rules in solving problems. In the designer/fabricator knowledge base, however, the global reasoning requirements of a given task are difficult to conceptualize as a series of linear local decisions, resulting in significant focus problems.

In order to resolve this problem a system architecture is needed which allows for the maintenance of multiple layers of contexts, goals and plans. One method of doing so might be to organize the information in the knowledge base

---

[9]Jarke and Vassiliou, "Coupling Expert Systems with Database Management Systems," pp. 72-73.

in a hierarchical fashion, where each node in the tree is a "specialist" which controls processing of rules within its domain.

The use of system specialists as a method for maintaining appropriate levels of abstraction has been implemented in classification systems and proposed for decision-support systems. Given that the underlying knowledge base can be viewed as a network of cause-effect links, classification specialists provide focus in the pursuit of correct causes, whereas decision-support specialists provide focus in the pursuit of correct effects. Embedded problem solving techniques in the two systems differ in a similar fashion.[10]

One way to veiw specialists in the designer/fabricator knowledge base would be as nodes in a hierarchical arrangement of superstructures and substructures. For example, "piers" and "foundations" are two substructures of the superstructure "bridge".

Given that specialists are determined by a hierarchical arrangement of structures, one method of implementation might be to embed control directives in taxonomic rules such as the following which represents a pier and its substructures:

```
pier :- substructure_1,
          :
        substructure_n.
```

This method represents a modification of Prolog's two-part system architecture, the consequences of which is a softening of the boundary between the inference engine and inference rules.

A second method is to use a combination of rules and frames where frames [Minsky, 1975] are used to represent the specialists and rules are used to

---

[10] B. Chandrasakaran, "Expert Systems: Matching Techniques to Tasks," in *Artificial Intelligence Applications for Business: Proceedings of the NYU Symposium, May, 1983,* ed. Walter Reitman, (Norwood, New Jersey: Ablex Publishing Corporation, 1984). p. 58.

represent inferential knowledge in the domain. Centaur uses this approach in a classification system for medical diagnosis. In Centaur control information is represented explicitly and separately from inferential knowledge, and is associated with specific slots in the specialist frame.[11]

The next section discusses modeling of inference rules within a hierarchical framework of specialists.

3.3.2 Modeling of Inferential Knowledge

Two other types of rules are important for structuring the remaining inferential knowledge in the knowledge base. The first type, transformation rules, describe change. Transformation rules interact with the taxonomic rules presented in the last subsection, providing a source of derivable facts. For example, if a pier and foundation are substructures of a bridge (taxonomic rule) and thermal effects create forces that act on rigidly clamped substructures and superstructures (transformation rule), then it can be inferred that a given pier will react in a calculable way to thermal effects.[12]

The second type of rule models the effect of state changes in one substructure on its superstructures. For example, a state change in "pier" may produce a change in the state of the superstructure "bridge". This type of rule has the form "*<STATE CHANGE IN SUBSTRUCTURE> causes <STATE CHANGE IN SUPERSTRUCTURE>*". All state change rules whose left-hand side deals with a given substructure are associated with the specialist for that substructure. Because of their hierarchical arrangement, specialists can determine the ef-

---

[11]Janice S. Aikins, "Prototypical Knowledge for Expert Systems," *Artificial Intelligence* 20:2 (1983): pp. 198.

[12]Hillman, "Knowledge-Based System for the Designer/Fabricator Interpreter," p. 4.

fect of a state change on the immediately larger structure of which it is a part and can call that structure's specialist with appropriate information. This process is repeated until state changes are propagated to the desired level of abstraction. The addition of a blackboard-like architecture [Erman et al, 1980] would make it possible to account for interaction between substructures in the model.[13]

[13]Chandrasakaran, "Expert Systems: Matching Techniques to Tasks," pp. 58-60.

Chapter 4
Database Models

4.1 Introduction

In Prolog, knowledge about a domain is represented by rules and facts. In general, rules are used to express definitions and to say that a fact depends upon a group of other facts. For example, a structure is a bridge if it has a foundation, piers and a number of other specific components.

In contrast, facts are represented in the knowledge base by simple assertions about objects and their relationships. For example, one such assertion might represent the length of a bridge currently under construction. The database in Figure 1-2, page 5, is comprised of all such assertions for the domain of the knowledge-based system.

The remainder of this chapter discusses the database component of the designer/fabricator decision-support system. Section 4.2 summarizes nonrelational approaches to database modeling. Section 4.3 deals with the structures, integrity constraints and data manipulation language of the relational model.

4.2 Nonrelational Database Models

Database models can be categorized according to the data structures and data manipulation operators they present to the user. The four major categories are:

- Inverted List
- Hierarchic
- Network

- Relational

The main difference between nonrelational and relational models is that the user of a relational system sees the data as tables and nothing but tables. In contrast, the user of an inverted list, hierarchic or network system sees other data structures in addition to or instead of tables.[14] The remainder of this section provides a brief description of the data structures and data manipulation operators for each of the three types of nonrelational database models.

4.2.1 Inverted List

An inverted list database consists of a collection of files or tables which are divided up into rows (records) and columns (fields). Unlike a relational database, rows in an inverted list database are ordered within and possibly between tables. Ordering of rows across all tables defines a total ordering for the database, referred to as the *database sequence.*

Unlike relational databases, inverted list databases allow the user to view certain access paths (in particular, certain indexes) in addition to tables. Indexes allow both direct and sequential access on the basis of search key values.

Data manipulation operators in inverted list databases are dependent on record addressing and fall into two categories:

1. Operators that determine the address of a record.

2. Operators that insert, update or delete a record once its address is known.[15]

---

[14]C. J. Date, *An Introduction to Database Systems: Volume I.* (Reading, Mass: Addison-Wesley Publishing Company, 1986), p. 21.

[15]Ibid., pp. 487-489.

4.2.2 Hierarchic

In the hierarchic database model, the entire database can be viewed as a tree, where the hierarchical sequence of the tree defines a total ordering for the set all all records in the database. This ordering applies not only to record types, but to occurrences of records of the same type as well. The principal difference between the hierarchic and relational model is that in a hierarchic database certain information that would be represented in a relational database by foreign keys (Section 4.3.1) is represented by parent-child links.

Data manipulation operators in a hierarchic database process data represented in the form of trees and are typically all record-level operators. Examples of the types of tasks performed by these operators include:

1. Locate a specific tree in the database.
2. Move from record to record within the tree.
3. Insert, update or delete a specified record.[16]

4.2.3 Network

The network model represents the last of the three categories of nonrelational systems. Like the hierarchic database model, the network model consists of parent and child records. In the network model, however, a given child record can have any number of parents. The database can be thought of more precisely as a set of record types, together with a set of link types, where each occurrence of a given link type consists of one occurrence of the parent record type and an ordered set of occurrences of the child record type.

Data manipulation operators in the network model process data in the

---

[16] Ibid., pp. 503-508.

form of records and links. Representative tasks include:

1. Locate a specific record given a value of a field in the record.

2. Move from a parent to a child, a child to another child, or a child to a parent in a given link.

3. Insert, update or delete a specified record.[17]

4.3 The Relational Model

The relational model developed as a way of shielding users of large data banks from potentially disruptive changes in data representation.[18] As mentioned in the introduction to Section 4.2, relational databases are viewed by the user as tables and nothing but tables, regardless of the way in which data is stored at the internal level. More precisely:

> A relational database is a time-varying collection of data, all of which can be accessed and updated as if they were organized as a collection of time-varying tabular (nonhierarchic) relations of assorted degrees defined on a given set of simple domains.[19]

In addition to data structures which support the preceeding definition, the relational model consists of integrity constraints and a relational algebra for data manipulation. The remainder of this section discusses these three components of the relational model in greater detail.

---

[17] Ibid., pp. 541-547.

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

[19] E. F. Codd, "Extending the Database Relational Model to Capture More Meaning," *ACM Transactions on Database Systems* 4:4 (December, 1979): p. 399.

4.3.1 Structures

Structurally, the relational model can be described in terms of its basic components (relations) and the way in which associations are formed between these components (primary and foreign keys).

RELATIONS. Roughly speaking, a relation corresponds to a table which consists of attributes (columns) and tuples (rows). Given that the term *domain* is defined as a set of values of similar type, a relation on domains *D1,D2,...,Dn* (not necessarily distinct) can be defined more precisely as consisting of the following two components:

1. *Heading:* A fixed set of attributes *A1,A2,...,An*, such that each attribute *Ai* corresponds to exactly one of the underlying domains *Di* (*i* = 1,2,...,*n*).

2. *Body:* A time-varying set of tuples, where each tuple consists of a set of attribute-value pairs (*Ai:vi*) (*i* = 1,2,...,*n*), one pair for each attribute *Ai* in the heading. For each pair, *vi* is a value from the unique domain *Di* that is associated with *Ai*.[20]

Given that all domains are simple (nondecomposable), a relation has a tabular representation with the following properties:

- There are no duplicate rows.
- Rows are unordered.
- Columns are unordered.
- All table entries are atomic (i.e., there are no repeating groups).[21]

PRIMARY AND FOREIGN KEYS. Primary keys provide the sole tuple-level addressing mechanism within the relational model. The primary key for a relation *R* is chosen from among one of the relation's n (n>=1) candidate keys.

---

[20]Date, *An Introduction to Database Systems: Volume I,* pp. 239-240.

[21]Codd, "Extending the Database Relational Model to Capture More Meaning," p. 399.

A candidate key is defined as a collection of attributes from *R* with the following time-independent properties:

- *UNIQUENESS:* No two tuples in *R* have the same K-component.
- *MINIMALITY:* No attribute of K can be dropped without destroying the uniqueness property.[22]

In the relational model, associations between relations are represented solely by values, rather than by structural links such as pointers. Specifically, references from one relation to another are accomplished through foreign-to-primary key matches, where a *foreign key* is an attribute or combination of attributes in a relation *R2* whose values are required to match those of the primary key of some relation *R1* (*R1* and *R2* not necessarily distinct).[23]

In addition to providing an addressing mechanism for the database and a means of forming associations between relations, primary keys play an important role in maintaining integrity constraints. This function is discussed in greater detail in the next subsection.

4.3.2 Integrity Constraints

Given that a "base relation" is an autonomous, named relation (i.e., it is not completely derivable from any other base relation(s)), inserts into, updates of, and deletions from base relations are constrained by the following rules:

1. *ENTITY INTEGRITY:* Attributes participating in the primary key of a base relation cannot accept null values.

2. *REFERENTIAL INTEGRITY:* If a base relation *R2* includes a foreign key *FK* which matches the primary key *PK* of some base relation *R1* (*R1* and *R2* not necessarily distinct), then every value of

---

[22]Ibid., p. 400.

[23]Date, *An Introduction to Database Systems: Volume I*, pp. 251.

FK in *R2* must be either:

- equal to the value of *PK* in some tuple of *R1*, or
- wholly null.[24]

4.3.3 Relational Algebra

The data manipulation language of the relational model consists of the following components:

1. A set of operators known as the *relational algebra.*
2. An *assignment operator* which assigns the result of an algebraic expression to some other relation (for example, $Z := X \cup Y$).

In relational systems the relational algebra and assignment operator are generally used to define the scope of an explicit insert, update or delete operation. Since these operations and the use of the assignment operator are self-explanatory, the remainder of this section is devoted to a discussion of the relational algebra.

The original relational algebra consists of four set operators and four special relational operators which are illustrated in Figure 4-1, page 25, and defined informally as follows:

- SET OPERATORS

  1. *Union:* resulting relation consists of all tuples appearing in one or both of two specified relations.
  2. *Intersection:* resulting relation consists of all tuples appearing in both of two specified relations.
  3. *Difference:* resulting relation consists of all tuples belonging to the first, but not to the second of two specified relations.

---

[24] Ibid., p. 252.

4. *Cartesian Product:* resulting relation consists of all possible pairs of concatenated tuples, one from each of two specified relations.[25]

- SPECIAL RELATIONAL OPERATORS

1. *Select:* resulting relation consists of a subset of the tuples (rows) of a specified relation.

2. *Project:* resulting relation consists of a subset of the attributes (columns) of a specified relation.

3. *Theta-Join:* resulting relation consists of all possible concatenated pairs of tuples, one from each of two specified relations such that for each pair the value of a given attribute in the first relation is related to the value of a given attribute defined on the same domain in the second relation in a specified way. When the values of the two attributes are equal the operator is known as *equi-join.* When the values of the two attributes are equal and redundant columns are removed from the resulting relation the operation is known as *natural join.*

4. *Divide:* resulting relation consists of all values of one attribute of a specified relation *R1* such that its Cartesian product with the sole attribute of a specified unary relation *R2* is included in *R1.*[26]

Note that the set operators union, intersection and divide apply only to pairs of union-compatible relations (i.e., relations with a one-to-one correspondence between attributes, with corresponding attributes defined on the same domain).

4.4 Discussion

Of the four categories of database models discussed the relational model is best suited for representing information in the designer/fabricator knowledge base. In addition to clearly-defined data manipulation operators and integrity

---

[25]Ibid., pp.257-258.

[26]Codd, "Extending the Relational Model to Capture More Meaning," pp. 400-403.

rules, the relational approach offers the following advantages:

- Relational databases have a natural affinity with the types of knowledge which need to be captured in the designer/fabricator knowledge base.
- Engineers feel comfortable working with tabular data representations.
- The viewpoints of designers and fabricators can be easily compared.
- Update requirements for managing designer/fabricator interactions can be accurately specified.
- The elicitation of each expert's knowledge can be accomplished in the same fashion and represented consistently.[27]

The next chapter discusses semantic extensions to the relational database model.

---

[27] Hillman, "Knowledge-Based System for the Designer/Fabricator Interpreter," pp. 4-6.

RELATIONS

R1(A B)

a	1
a	2
b	2
c	1
c	3

R2(C D)

a	1
b	1
b	2
c	3

R3(E)

1
3

SET OPERATORS

UNION
R1 ∪ R2(F G)

a	1
a	2
b	1
b	2
c	1
c	3

INTERSECTION
R1 ∩ R2(F G)

a	1
b	2
c	3

DIFFERENCE
R1 - R2(A B)

a	2
c	1

CARTESIAN PRODUCT
R1 X R2(A B C D)

a	1	a	1
a	1	b	1
a	1	b	2
a	1	c	3
a	2	a	1
a	2	b	1
a	2	b	2
a	2	c	3
b	2	a	1
b	2	b	1
b	2	b	2
b	2	c	3
c	1	a	1
c	1	b	1
c	1	b	2
c	1	c	3
c	3	a	1
c	3	b	1
c	3	b	2
c	3	c	3

SPECIAL RELATIONAL OPERATORS

SELECT
R1(A=a)(A B)

a	1
a	2

PROJECT
R1(A)

a
b
c

THETA-JOIN
R1[B>E]R3(A B E)

a	2	1
b	2	1
c	3	1

EQUI-JOIN
R1[B=E]R3(A B C)

a	1	1
c	1	1
c	3	3

NATURAL-JOIN
R1[B*C]R2(A B)

a	1
c	1
c	3

DIVIDE
R1[B÷E]R3(A)

c

Figure 4-1: Relational Operators

Chapter 5
Semantic Modeling

5.1 Introduction

Semantic modeling refers to the process of incorporating more meaning into the database. In general, semantic models consist of the following four sets:

1. semantic concepts which are used to talk about the real world.
2. symbolic objects which represent the semantic concepts.
3. operators which manipulate the symbolic objects.
4. integrity constraints.[28]

The relational model described in the last section captures a limited amount of semantic information. For example, foreign-to-primary key matches provide some information about the meaning of a particular relation. The remainder of this chapter describes two database models which extend the semantic capabilities of the relational model, the Entity-Relationship Model and the Extended Relational Model RM/T. Both models are discussed in terms of their semantic concepts, symbolic objects, data manipulation operators and integrity constraints.

5.2 The Entity-Relationship Model

5.2.1 Semantic Concepts

The entity-relationship model, as propsed by Chen (1976), is a generalization or extension of, rather than an alternative to the relational database model. The model views the world as consisting of entities and relationships. An *entity*

[28] Date, *An Introduction to Database Systems: Volume I*, p. 610.

is defined as a thing which can be distinctly identified. Examples of entities include a specific person, company or event. A *relationship* is defined as an association between entities. Examples of relationships include PROJECT-MANAGER and FATHER-SON.

5.2.2 Symbolic Objects

ENTITY RELATIONS. The entity-relationship model separates information about entities from information about relations. Thus, entities are represented by *entity relations* and relationships by *relationship relations.*

Figure 5-1, page 28, illustrates an entity relation in tabular form. As in the relational database model:

- Each table has a unique identifier or primary key.
- Each row of the table is a tuple (in this case, an entity tuple) of attribute-value pairs.
- There are no duplicate rows.
- Rows are unordered.
- Columns are unordered.
- All table entries are atomic.

The main difference between Figure 5-1 and a table in the relational model lies in the column headings. In the relational model columns are headed by attributes, each of which are defined on a specific domain. In the entity-relationship model, columns are headed by attributes and value sets. A few definitions are required in order to understand this distinction.

Entities, such as those represented by the primary key EMPLOYEE-NO in Figure 5-1, are grouped into *entity sets*, which are not necessarily mutually disjoint. For example, the individual represented by EMPLOYEE-NO 2566 may

		---Primary--- Key					
Attribute		Employee-No.	Name		Alternative-Name		Age
Value Set (Domain)		Employee-No.	First-Name	Last-Name	First-Name	Last-Name	No.-of-Years
Entity		2566	Peter	Jones	Sam	Jones	25
		3378	Mary	Chen	Barb	Chen	23
		.	.	.	.	.	.
		.	.	.	.	.	.

Figure 5-1: Regular Entity Relation EMPLOYEE[29]

belong to both the entity sets PERSON and MALE-PERSON. Information about entities is represented by attribute/value pairs such as COLOR/RED and NAME/JOHN_DOE. Values such as RED, JOHN and DOE are grouped into *value sets* such as COLOR, FIRST-NAME and LAST-NAME. An *attribute* of an entity relation is defined as a function which maps from an entity set into such a value set or Cartesian product of value sets. Thus the concept of value set is similar to that of domain in the relational model, except that value sets form an integral part of the tabular representation of data, thereby making semantic information more accessible to the user.

Similar properties apply to relationship relations with the following distinctions:

- Each row of the table is a *relationship tuple.*
- The primary key is represented by the primary keys of the involved entities.
- The *role* of an entity in a relationship is its function in that

[29] Peter Pin-Shan Chen, "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Transactions on Database Systems* 1:1 (March, 1976): pp. 17.

- *Relationship attributes* are attributes of the relationship other than entity attributes involved in the primary key.

Finally, the entity relationship model distinguishes between weak and regular relations. Regular entity and relationship relations were discussed in the preceeding paragraphs. A relation is a *weak entity relation* when a relationship must be used to uniquely identify an entity in an entity set. For example, dependents might be identified in the database by their names and the value of the primary key of the employee supporting them. Likewise, if one or more entities involved in a relationship relation are identified by other relationships, the relation is a *weak relationship relation.*[30]

5.2.3 Data Manipulation Language

Chen states that "information requests may be expressed using set notions and set operations";[31] however, his article does not clearly specify which operators are supported. Although there is apparently no union or explicit join, the operators in the entity-relationship model appear to be basically the same as in the relational algebra. As in the relational model, these operators are used to define the scope of explicit insert, update or delete operations.

5.2.4 Integrity Constraints

The entity-relationship model supports the following integrity "rules":

- When deleting an entity tuple, delete any entity tuple whose existence depends on it, and delete relationship tuples which are associated with the entity.

- When inserting an entity tuple in an entity relation, check to see

---

[30]Peter Pin-Shan Chen, "The Entity-Relationship Model - Toward A Unified View of Data," *ACM Transactions on Database Systems* 1:1 (March, 1976): pp. 10-18.

[31]Ibid., p. 25.

that the entity primary key does not already exist and is an acceptable value.

- When inserting a relationship tuple in a relationship relation, verify the existence of the entities whose primary keys comprise the relationship primary key.

- When updating a value which is part of an entity primary key, cascade the update to all relationship and entity relations which use this value as a part of their primary keys.

- When inserting or updating any value, check to see that the value is acceptable.

The last integrity rule requires maintenance of the following:

- constraints on allowable values for a value set.

- constraints on permitted values for a certain attribute (i.e., certain allowable values may not be permitted for a given attribute in a given relation.)

- constraints on existing values in the database, including:

    - constraints between sets of existing values (for example, the entity set MANAGER must be a subset of the entity set EMPLOYEE).

    - constraints between particular values (for example, the value which represents an individual's tax must be less than the value which represents that person's salary).[32]

Chen's paper does not explicitly define foreign key rules. However, the user can specify that a given relation is one-to-one, one-to-many or many-to-many, in which case certain foreign key rules are implicitly understood.[33]

---

[32]Ibid, pp.22 -25.

[33]Date, *An Introduction to Database Systems: Volume I*, p. 612.

5.2.5 Entity-Relationship Diagrams

The entity-relationship model offers a diagrammatic technique for representing the entities and relationships described in this section. Figure 5-4, page 32, adapted from Chen's article, illustrates the principal characteristics of entity-relationship diagrams:

- Each entity set is represented by a rectangular box. A double rectangular box (not depicted) can be used to represent a weak entity relation.

- Each relationship set is represented by a diamond-shaped box which is connected by lines to the entity sets participating in the relationship. Note that a relationship set may be defined on one or many entity sets.

- Lines on the diagram are labeled *1*, *m* or *n*, where *1:1* indicates a one-to-one, *1:n* indicates a one-to-many and *m:n* indicates a many-to-many mapping.[34]

[34]Chen, "The Entity-Relationship Model - Toward a Unified View of Data," pp. 19-20.

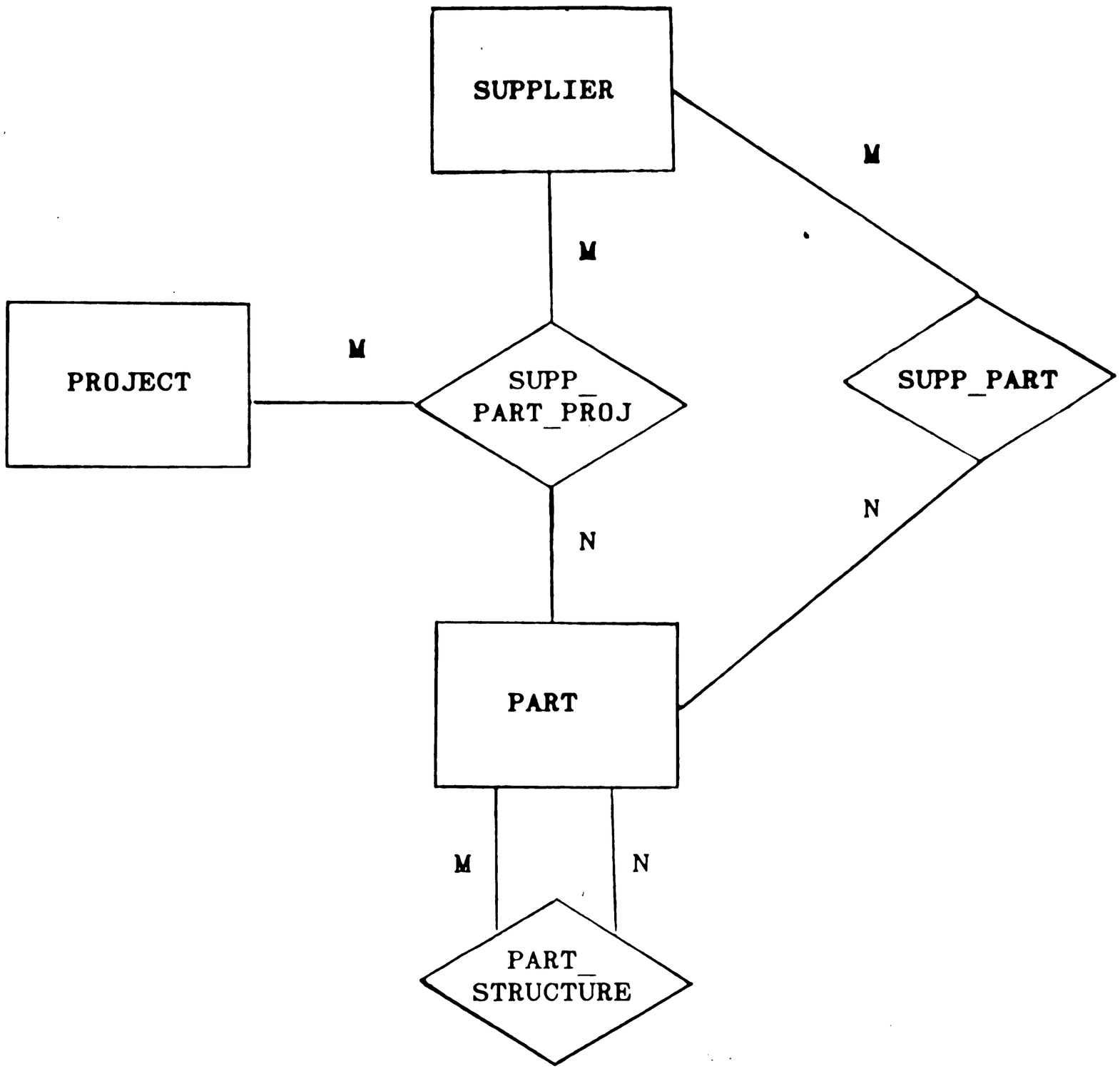

Figure 5-2: Example Entity Relationship Diagram

5.3 The Extended Relational Model RM/T

5.3.1 Semantic Concepts

The extended relational model RM/T was introduced by Codd in 1979. The model views the real world as consisting of entities, where the concept of entity encompassses both entities and relationships as defined in the entity-relationship model.

5.3.2 Symbolic Objects

SURROGATES. In RM/T user-defined primary keys (and associated foreign keys) are replaced by system-controlled entity representatives which are associated with specific domains. All internal system references are accomplished using these *surrogates*, even though the user may still address the system via user-defined primary keys. The following properties apply to surrogates:

- Each surrogate is unique and guaranteed never to change.
- Two surrogates are equal if and only if they denote the same entity.

One of the domains of the database, the *E-Domain* serves as the source of all surrogates. Attributes defined on this domain are known as *E-Attributes* and are given names which end in the special character ¢.

E-RELATIONS. Entities are represented in RM/T by E-Relations and P-Relations. An *E-Relation* is a unary relation which lists the surrogates for all entities of a given type which currently exist in the database. The name of the E-Relation is the name of the entity type. The name of the E-Relation's single E-Attribute consists of the name of the E-Relation concatenated with the special character ¢. An example E-Relation, EMPLOYEE, follows:

EMPLOYEE¢
alpha
beta

P-RELATIONS. P-Relations are used to represent properties of entities. The name of the E-Attribute which serves as the primary key consists of the name of the entity type with a trailing ¢. Values for the E-Attribute represent the surrogates for all of the entities in the P-Relation. The remaining at-

tributes represent properties of the entity type. All such properties may be grouped into one P-Relation or divided into many P-Relations at the discretion of the database designer.[35] An example P-Relation, EMPL-NAME, follows:

EMPLOYEE	LAST-NAME	FIRST-NAME
alpha	Doe	John
beta	Smith	Mary

5.3.3 Data Manipulation Language

The data manipulation language for RM/T consists of the following:

1. The relational algebra.
2. The assignment operator.
3. Operators which manipulate various RM/T objects.

The relational algebra and assignment operators were discussed in Section 4.3.3. The special RM/T operators are:

- NAME OPERATORS
    1. *NOTE:* Returns the character string representation of a relation or null.
    2. *TAG:* Returns the Cartesian products of a relation with the character string representation of the relation.
    3. *DENOTE:* Returns the relation denoted by a particular character string representation.
- SET OPERATORS
    1. *COMPRESS:* Given that f is an associative and commutative operator that maps a pair of relations into a relation, and Z is a set of relations such that f can be validly applied to every

[35] Codd, "Extending the Relational Model to Capture More Meaning," pp. 409-413.

pair of relations in Z, then COMPRESS(f,Z) is the relation obtained by repeated pairwise application of f to the relations in Z.

2. *APPLY:* Creates a set of all relations $f(z)$ such that f denotes a unary operator that maps relations into relations, Z is a set of relations which are not necessarily union compatible, and z is a member of Z. *APPLY* allows every member of a set of relations to be evaluated in an algebraic equation in any place where a relation name would be syntactically valid.

3. *Partition by Attribute (PATT:)* Creates a set of relations obtained by partitioning a relation R per all the distinct values of an attribute A. (Note, R = UNION/PATT(R,A))

4. *Partition by Tuple (PTUPLE:)* Creates a set of relations obtained by promoting each tuple of a relation R into a single-tuple relation. (Note, R = UNION/PTUPLE(R))

5. *Partition by Relation (PREL:)* Creates a set of relations whose only member is the relation R. (Note, R = UNION/PREL(R))

6. *SETREL:* Creates a set of relations from any number of explicitly named relations. For example, SETREL(*R1*, *R2*,...,*Rn*) creates a set containing relation *R1* through *Rn*.

RM/T also provides three graph operators, *OPEN*, *CLOSE* and *STEP*, the details of which are beyond the scope of this work.[36]

5.3.4 Integrity Constraints

CLASSIFICATION OF ENTITIES. There are three categories of entities in RM/T:

1. *Characteristic entities* fill a subordinate role in describing entities of some other type. (Characteristic entities are existence-dependent on the entities they describe.)

2. *Associative entities* fill a superordinate role by interrelating entities.

3. *Kernel entities* are entities which are neither characteristic or associative.

---

[36] Ibid., pp. 425-428.

TYPE HIERARCHIES. A given entity may be of several types simultaneously. An entity type and its subtypes constitute a type hierarchy with the following characteristics:

- Properties of a supertype apply to all of its subtypes.
- All associations in which a supertype participates are automatically associations in which a subtype participates.[37]

INTEGRITY RULES. RM/T maintains a formal catalog structure which identifies entities as characteristic, associative or kernel and maintains information about type hierarchies. This information is used to enforce the model's integrity rules:

1. *ENTITY INTEGRITY:* (Section 4.3.2)
2. *REFERENTIAL INTEGRITY:* (Section 4.3.2)
3. *ENTITY INTEGRITY IN RM/T:* E-Relations cannot be updated or accept null values. Insertions and deletions are allowed.
4. *PROPERTY INTEGRITY:* A surrogate appearing in a P-relation must also appear in the E-Relation for that entity type.
5. *CHARACTERISTIC INTEGRITY:* In order for a characteristic entity to exist in the database, the entity which it describes most immediately must also exist in the database.
6. *ASSOCIATIVE INTEGRITY:* Assuming there are no explicit integrity constraints to the contrary, an associative entity can exist in the database even though one or more entities in the association are unknown. (The surrogate *E-null* is used to indicate that a participating entity is unknown.)
7. *SUBTYPE INTEGRITY:* A surrogate belonging to the E-Relation for an entity of type *e* must also belong to the E-Relations of entity types of which *e* is a subtype.[38]

---

[37] Date, *An Introduction to Database Systems: Volume I*, p. 619.

[38] Codd, "Extending the Relational Model to Capture More Meading," pp. 411 - 421.

Chapter 6
Conclusions

Of the database models discussed in this work (relational, inverted list, network and hierarchic), the relational model is best suited for the designer/fabricator knowledge base because of its clearly-defined data manipulation operators and integrity rules, tabular representation of data, and natural affinity with the types of information which need to be incorporated in the knowledge base.

Semantic extensions to the relational model are summarized in the following table:

	ENTITY-RELATIONSHIP	RM/T
SEMANTIC CONCEPTS	entities relationships	entities
SYMBOLIC OBJECTS	entity relations relationship relations	E-Relations P-Relations
OPERATORS	relational algebra (?)	relational algebra assignment operator RM/T operators
INTEGRITY RULES	relational integrity rules (?)	integrity rules: entity referential entity in RM/T property characteristic association subtype

The extended relational model RM/T offers the following advantages over the entity-relationship model:

1. The distinction between entitites and relationships is not always clear

in the real world. Such ambiguities are resolved by classifying everything in the real world as an entity.

2. The use of system-defined surrogates eliminates the disadvantages inherent in user-defined primary keys.

3. RM/T provides a clearly-defined data manipulation language which includes the relational algebra, assignment operator and special RM/T operators.

4. RM/T provides a clearly-defined set of integrity rules in addition to the integrity rules in the relational model.

RM/T does not provide an alternative to the diagrammatic technique offered by the entity-relationship model. One solution to this problem is to adapt the entity-relationship diagram to the requirements of RM/T as follows:

- Each kernel entity type would be represented as a rectangle.
- Each associative entity type would be represented by a diamond-shaped box.
- Each characteristic entity type would be represented by a double rectangle.

Further research is required to determine whether the concept of a hierarchical arrangement of specialists is applicable to the designer/fabricator knowledge base. If it is, then grouping information in the knowledge base according to specialists provides a framework for acquiring information from the experts whose knowledge will be encoded in the system.

The process of acquiring and modeling information within this framework is outlined in Figure 6-1, page 40. The first stage of the process is to work with the expert to determine a preliminary version of the specialist tree. The information in the tree corresponds to the type of information typically contained in taxonomic rules. Once a tree of height of N has been built, an iterative process of evaluating database facts and transformation rules for the bottom

two levels (N and N-1) of the tree begins.

When this process is complete, state change rules, which model the effects of changes in level N-1 on level N, can be modeled. Once all database facts, transformation rules and state change rules seem correct, the value of N is decremented. If N>0, the new level N-1 is evaluated in a similar fashion. At any stage in the process previous levels of the tree can be modified, provided that changes are systematically propagated up the tree. When N=0 all levels, including the root, have been evaluated, and the process is complete.

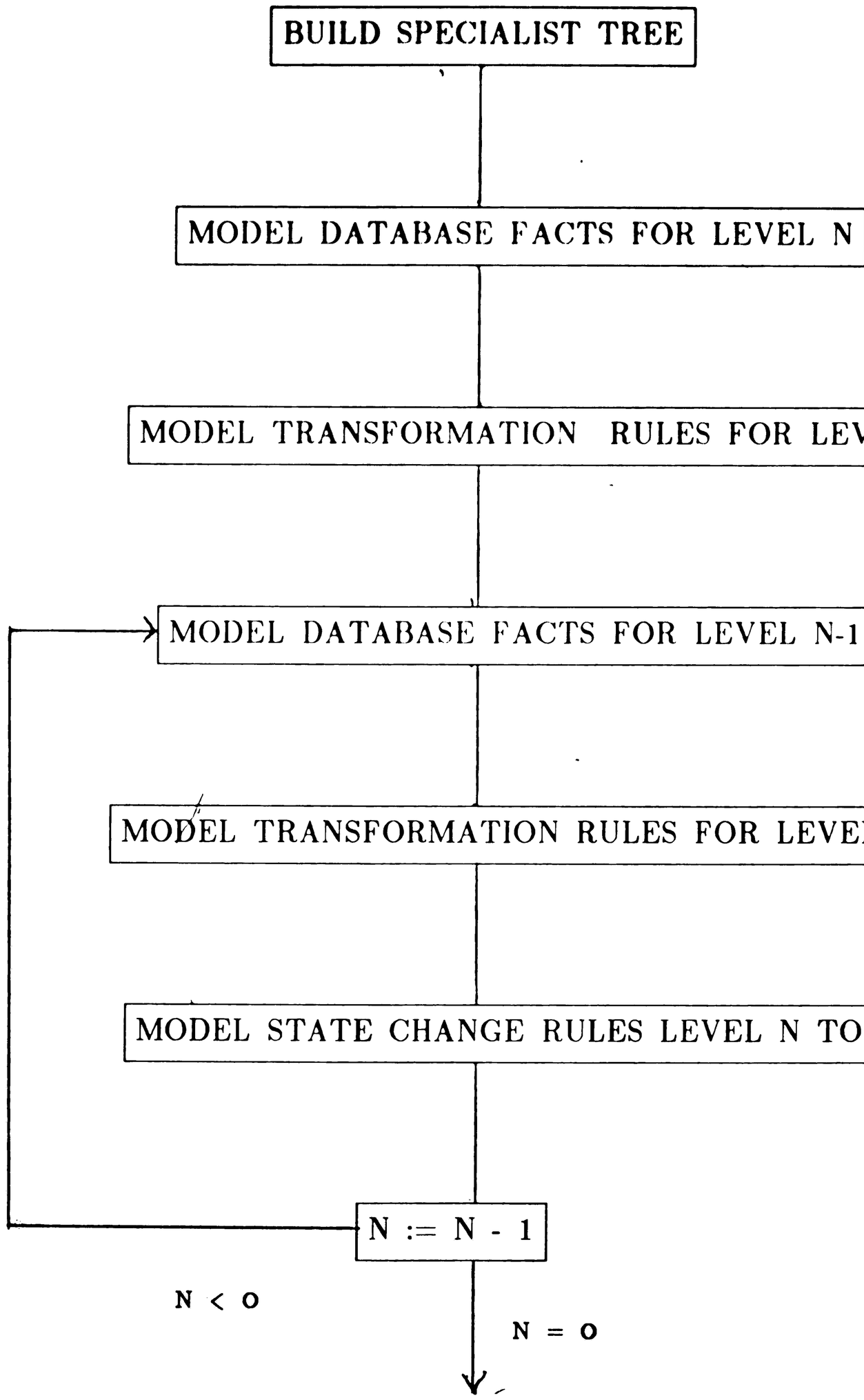

Figure 6-1: Model for Conceptual Analysis

Bibliography

Aikins, Janice S. "Prototypical Knowledge for Expert Systems." *Artificial Intelligence* 20:2 (1983): 163-210.

Chandrasekaran, B. "Expert Systems: Matching Techniques to Tasks." in *Artificial Intelligence Applications for Business: Proceedings of the NYU Symposium, May 1983*, pp. 41-64. Edited by Walter Reitman. Norwood, New Jersey: Ablex Publishing Corporation, 1984.

Chen, Peter Pin-Shan. "The Entity-Relationship Model - Toward a Unified View of Data." *ACM Transactions on Database Systems* 1:1 (1976): 9-36.

Clocksin, W. F. and Mellish, C.S. *Programming in Prolog.* Berlin: Springer-Verlag, 1984.

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

Codd, E. F. "Extending the Relational Model to Capture More Meaning." *ACM Transactions on Database Systems* 4:4 (1979): 397-434.

Date, C. J. *An Introduction to Database Systems: Volume I.* Reading, Mass.: Addision-Wesley Publishing Company, 1986.

Erman, L. D., Hayes-Roth, F., Lesser, V. R. and Reddy, P. R. "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty." *ACM Computing Surveys* 12:2 (1980): 213-253.

Feigenbaum, Edward A. "The Art of Artificial Intelligence." *Proc. IJCAI-77* (1977): 1014-1029.

Futo, I., Darvas, F., and Szeredi, P. "The Application of Prolog to the Development of QA and DBM Systems." in *Logic and Databases*, pp. 347-376. Edited by Herve Gallair and Jack Minker. New York: Plenum Press, 1978.

Hayes-Roth, Frederick, Waterman, Donald A. and Lenat, Douglad, B., eds. *Building Expert Systems.* London: Addison-Wesley Publishers, 1983.

Hillman, Donald. "Knowledge-Based System for the Designer/Fabricator Interface." Unpublished Manuscript, Lehigh University, Bethlehem, PA, 1986.

Jarke, Matthias and Vassiliou, Yannis. "Coupling Expert Systems with Database Management Systems." in *Artificial Intelligence Applications for Business: Proceedings of the NYU Symposium, May 1983*, pp. 65-85. Edited by Walter Reitman. Norwood, New Jersey: Ablex Publishing Corporation, 1984.

Minsky, M. "A Framework for Representing Knowledge." in *The Psychology of Computer Vision*, Edited by P. H. Winston. New York: McGraw Hill,

1975.

Rauch-Hindin, Wendy B. *Artificial Intelligence in Business, Science, and Industry: Volume II - Applications.* Englewood Cliffs, New Jersey: Prentice- Hall, Inc., 1985.

Robinson, J. A. "A Machine Oriented Logic Based on the Resolution Principle," *JACM* 1:4 (1965): 23-41.

Sowa, John F. *Conceptual Structures: Information Processing in Mind and Machine.* Reading, Mass: Addison-Weslely Publisher, 1984.

Vita

The author was born in Tamaqua, Pennsylvania, on July 16, 1955, the first child of Michael and Grace Pollak. She graduated with honors from Tamaqua Area High School in 1973, and Magna Cum Laude from Muhlenberg College, Allentown, Pennsylvania, in 1984 with an B.A. in Psychology. While at Muhlenberg she was chosen to be a member of Phi Beta Kappa and the National Honor Society in Psychology.

After being accepted as a candidate for the degree of Master of Science in 1985, the author was awarded a one-year graduate assistantship managing a research database for the Department of Economics at Lehigh University. In the summer of 1986 she was employed by the Management Information Systems Department at Air Products and Chemicals, Inc., Trexlertown, Pennsylvania. In her final year of graduate study the author was appointed as a half-time teaching assistant in the Computer Science Department at Lehigh University.