DESIGN AND CONSTRUCTION OF MAINTAINABLE KNOWLEDGE BASES THROUGH EFFECTIVE USE OF ENTITY-RELATIONSHIP MODELING TECHNIQUES

by

WILLIAM YANCEY PIKE
B.S., University of West Florida, 1987

THESIS

Submitted in partial fulfillment of the requirements
for the degree of
Master of Science
College of Engineering
University of Central Florida
Orlando, Florida

Summer Term
1993

ABSTRACT

The use of an accepted logical database design tool, Entity-Relationship Diagrams (E-RD), is explored as a method by which conceptual and pseudo-conceptual knowledge bases may be designed. Extensions to Peter Chen's classic E-RD method which can model knowledge structures used by knowledge-based applications are explored.

The use of E-RDs to design knowledge bases is proposed as a two-stage process. In the first stage, an E-RD, termed the Essential E-RD, is developed of the realm of the problem or enterprise being modeled. The Essential E-RD is completely independent of any knowledge representation model (KRM) and is intended for the understanding of the underlying conceptual entities and relationships in the domain of interest. The second stage of the proposed design process consists of expanding the Essential E-RD. The resulting E-RD, termed the Implementation E-RD, is a network of E-RD-modeled KRM constructs and will provide a method by which the proper KRM may be chosen and the knowledge base may be maintained. In some cases, the constructs of the Implementation E-RD may be mapped directly to a physical knowledge base.

Using the proposed design tool will aid in both the development of the knowledge base and its maintenance. The need for building maintainable knowledge bases and problems often encountered during knowledge base construction will be explored.

A case study is presented in which this tool is used to design a knowledge base. Problems avoided by the use of this method are highlighted, as are advantages the method presents to the maintenance of the knowledge base. Finally, a critique of the ramifications of this research is presented, as well as needs for future research.

ACKNOWLEDGEMENTS

The author would like to thank his wife, Tracey, for inspiring, motivating, and occassionally forcing him to continue his education. The author offers no small amount of gratitude to Dr. Soheil Khajenoori for displaying tremendous amounts of patience; to Drs. Tom Peeples and Allan Lang for their assistance and support; and to his parents, Billy and Eola Pike, for their love and guidance through the years. Finally, but most importantly, the author would like to thank the Lord, without whose supreme assistance this work would be impossible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Although the topic of knowledge base and database integration has recently been an area of considerable research from both from academia and industry, for the most part this research has failed to integrate conceptual database design principles into the design of knowledge bases.

The need for such a design methodology in the knowledge base system world is inarguably a real one. The design of knowledge-based systems (KBS) and their underlying knowledge base management systems (KBMS) suffers from a lack of a de facto standard methodology (**Gonzalez and Dankel 1993**). This lack of a methodology can lead to a *paradigm shift*, in which, during the development of the KBS, the developer must shift to a new technology. (**Gonzalez and Dankel 1993**) This paradigm shift is caused when the initial selection of knowledge representation model (KRM) can not adequately perform its intended function. This represents perhaps the most serious problem in KBS development. However, there are other inherent problems, as described in (**Gonzalez and Dankel 1993**). One problem lies in the difference between solving traditional information-system problems and heuristic-oriented problems. The data needed for algorithmic problems can be determined fairly easily, while in the case of knowledge-based systems, sometimes the "nature and quantity" of the knowledge isn't known even by the experts. The process of knowledge acquisition can thus prove to be fairly frustrating. One of the underlying reasons for this, claims Earl Cox, a columnist for *AI Expert*, is a common perception that AI is commonly defined in "terms of ever more advanced knowledge representation schemes devoid and divorced from fundamental architectural and design

considerations." **(Cox 1993)** The lack of any recognized correlation between AI and conventional systems has lead to "confusion in aims and directions" of AI in the marketplace **(Cox 1993)**. Clearly, there is a need to apply sound, established traditional software development principles to AI system development.

The attitudes and mindsets of KBS developers are perhaps part of the problem. The roots of database research lie primarily in the "commercial sector's need for efficient and secure data processing systems." **(Jelly and Gray 1992)** Free from this requirement which would restrict research to mostly commercial applications, early KBS researchers developed an almost "renegade" approach to application development. Indeed, as Cox has pointed out, "there does seem to be a general consensus among knowledge engineers that AI is somehow completely removed from computer science, systems design, and functional decomposition." **(Cox 1993)**

Another viewpoint of this problem is stated in **(Cohen 1990)**. Paul Cohen blames much of the problem on the lack of qualitative vice quantitative research in AI. He states, "Much work is unevaluated and most evaluations are limited to measures of performance. System design appears arbitrary and, when justifications do appear, they are informal...Evaluation tends to be limited to performance evaluation, instead of tests of hypotheses of how behavior arises from the interaction of agents' architectures and their environments." Cohen goes on to describe what he terms the "strip mining" view of AI research. "AI researchers trash the space of questions about intelligence in much the same way that slash-and-burn cultures trash the rain forest. Both make very inefficient use of resources." As an example of "strip mining," Cohen points out the following: "The statement 'X is sufficient to produce Y' alleges but does not model or explain the alleged causal relationship between X and Y . . . Demonstrating that X is sufficient to produce Y does not show that X is a particularly good way to produce Y, or that X is necessary to

produce Y." This problem is very similar to the "nature and quantity" dilemma discussed above.

Quality has become somewhat of a buzzword in industry (e.g., "Total Quality Management".) As knowledge-based systems in specific and AI systems in general come out of the research lab and into the mainstream of the marketplace, the quality of these systems must be taken into consideration. It is pointed out in **(Fenn and Veren 1991)** that "adherence to a software engineering methodology and development lifecycle can significantly improve the quality of a completed system." Earl Cox has stated that "successful AI projects combine quality with concerns for economical solutions." **(Cox 1993)**

As the maintenance portion of any software project life cycle has historically been the costliest, a design technique should provide for maintenance in order to supply quality to the project **(Ignizio 1991)**, **(Parsaye and Chignell 1988)**, **(Debenham 1992)**.

To a great extent, these same problems or similar concerns can be seen in traditional database application development efforts. Semantic data models have been used as a design tool to solve these problems. Of all the semantic data models, Peter Chen's entity-relationship (E-R) model has become the most popular, due to a great extent to the popularity of the E-R diagram (E-RD), a graphical companion to the E-R model. **(Date 1990)**

Applying Peter Chen's classic E-R diagramming technique, or some variation thereof, to the design of a knowledge base (regardless of the knowledge representation technique used by the KBS) provides the developer with a proven methodology to ensure a more intelligent design. By developing E-RDs early in the development life-cycle of the KBS, designers can avoid the knowledge representation paradigm shift by determining the proper representation *a priori* implementation. Having a well-defined E-RD of the

knowledge base can also aid in maintaining the KBS. The effects of adding new knowledge or modifying existing knowledge can quickly be determined by consulting an E-RD.

This thesis proposes the use of entity-relationship diagrams as a design tool for the development of knowledge base systems. More specifically, a two-stage process is proposed in which a traditional E-RD, called the Essential E-RD, is developed based on the conceptual knowledge base as the first stage. This E-RD serves to identify the conceptual entities and relationships of the knowledge realm. In the second stage, the first E-RD is expanded to model the knowledge structures via extended E-RD structures. The resulting E-RD is called the Implementation E-RD. These E-RD structures, for the most part, have already been proposed in earlier bodies of research as development aids for DBS applications, although additional structures are proposed herein to better model knowledge concepts.

The use of a semantic data model is defended as a combination of the latter two levels of the three-level of integration of databases and knowledge bases. The first level, the physical layer, involves utilization of database management systems (DBMSes) to physically store the knowledge of a KBS, and the integration of traditionally KBS-oriented features into DBMSes. The second level, termed the pseudo-conceptual layer, starts to apply conceptual DBMS design methodologies into the design of a knowledge base. In this layer, the design is presented for a certain KRM only. At the conceptual layer, database design techniques are proposed for the design of the conceptual knowledge base, independently of a specific KRM.

As knowledge bases continue to grow, they will undoubtedly require a great deal of support from databases. Many expert-system shells now offer front-ends to popular database engines. Likewise, as database applications become more complex, they will

require intelligent features from knowledge based systems. An example of this is ongoing research in the database community of implementing *business rules* into databases and database applications. These rules, defined as "constraint(s) placed upon the business" (**Moriarty 1993**), have a five-stage design process very similar to the design process of KBSes. A reason expert systems fail is that they aren't integrated into the corporate computing architecture. "A high percentage of expert system programs result in a successful prototype from a technical point of view but fail to produce a system which is integrated into an organization's mainstream operational environment." (**Fenn and Veren 1991**). The corollary of this statement may also well be true; that is, the knowledge bases of intelligent systems are not being utilized by the "mainstream" corporate applications. This work serves as an important step in bringing the two camps together.

LITERATURE REVIEW

Research for this thesis was necessarily performed from two separate but complimentary viewpoints. Experts from both the database realm and the knowledge base realm have written extensively on issues similar to the ideas proposed herein.

Databases and knowledge bases share many similarities. Both serve to store the data necessary to make their respective systems perform. Both have established physical structures designed to optimize the retrieval of that data. Both have certain relationships between their logical design and their physical design. The union of databases and knowledge bases can be divided into three levels: the *physical* level, the *pseudo-conceptual* level, and the *conceptual* level. While the conceptual and the pseudo-conceptual levels are the primary concern of this paper, a review of all three levels will help establish a better baseline for the main premise to be presented later.

The Physical Level

The physical level represents the lowest level of abstraction in the integration of databases and knowledge bases. At the physical level, research has focused on many areas. Those areas discussed here will consist of:

- Storing/retrieving knowledge in/from a database management system (DBMS),
- Adding traditional expert system features to DBMSes, and
- Interfacing database systems (DBSes) and KBSes.

Frank Anger, Rita Rodriguez and Douglas Dankel have co-authored a series of papers on organizing expert systems' knowledge bases using databases and database design techniques. They have proposed utilizing a commercial relational DBMS

(RDBMS) to store the rules of an expert system's knowledge base (**Rodriguez et. al. 1989**). Their proposal calls for three RDBMS relations, or tables, to implement the knowledge base. The first, named **IF**, consists of the fields *rule#*, *assrt#*, and *assrtdescr*. The second table, **THEN**, is also made up of the fields *rule#*, *assrt#*, and *assrtdescr*. To track confidence, an integral part of rule-based systems, the table **RULE-CONF** is defined to consist of the fields *rule#* and *conf*. These fields are described as such:

rule# - a unique identifier of the rule
assrt# - a unique identifier of an assertion
assrtdescr - the textual description of the assertion
conf - a number which represents the confidence in the deduction

In this design, both the **IF** and **THEN** tables have a composite primary key consisting of the fields *rule#* and *assrt#*, while **RULE-CONF** uses *rule#* as its primary key (**Rodriguez et. al. 1989**).

The same paper also details the addition of procedural knowledge via a *trigger* relation. This table, called **TRIGGERS**, includes as a foreign key the field *assrt#*. When the inference engine fires a rule which involves assertion N, the system queries **TRIGGERS** to determine whether any procedures are to be invoked. An additional table, **PROCEDURES** (whose primary key *pname* is also a foreign key in **TRIGGERS**), contains the action to be performed (**Rodriguez et. al. 1989**) (**Anger et. al. 1988**).

An additional step in this direction has been proposed in (**Ito 1991**). Ito proposes a coupling of KBSes and DBMSes. Since the reconstruction of an existing database to perform the task of knowledge base manager is "burdensome", Ito suggests the knowledge representation scheme (KRS) provide the mechanisms required for coupling. Called IKD (Interface for integrating a Knowledge-based system and a Database system), the system serves as the interface between a KRS called KBUS and a relational database.

KBUS is composed of a frame-based system called FKBUS and a production system called PKBUS, in addition to IKD. FKBUS consists of several frames and sub-frames which include, among other items, actual SQL (Structured Query Language) code to retrieve knowledge from the database. In summary, Ito's paper proposes a knowledge-based system which uses a frame-based subsystem to retrieve knowledge from an SQL-compliant relational database.

Levent Orman of Cornell University proposed in **(Orman 1992)** that a three-layer abstraction ("external", "conceptual" and "internal" layers) of knowledge bases be developed, with each layer targeted to a specific user type. At what Orman calls the "internal level", targeted to system implementers, rules are to be "viewed as data." An interesting point of Orman's proposal is the case he makes for hierarchical databases to store rules, as opposed to the relational database approach championed in **(Anger et. al. 1988)**, **(Rodriguez et. al. 1989)** and **(Ito 1991)**. As a discussion of which database model is most suited for the storage, retrieval and management of knowledge constructs is beyond the scope of this paper, the point is simply made that **(Orman 1992)** provides a strong case for the physical level of abstraction of database/knowledge base integration.

Industry has also contributed to the physical level of DBMS/KBS unions. Many relational databases now supply *triggers*, which supply a primitive method of supplying rule-based processing. A trigger is defined to be invoked on a certain action or condition (cf., *trigger relations*, **(Rodriguez et. al. 1989)** **(Anger et. al. 1988)**). Unfortunately, triggers generally must be written in SQL, which doesn't provide the flexibility required to add true intelligence to a database. Sybase, Inc., an innovator in client-server RDBMS engines, has included the capability for "stored procedures" which can add a further level of intelligence to a database by defining certain processing to occur based on user-defined events. These stored procedures, which are compiled and execute on the server side of

database applications, allow more efficient processing than triggers. The influx of client-server database engines has provided another opportunity for DBMS/KBS unions. A query can be passed through a KBS on the client side before issuing the SQL code to the server side. Ingress, the relational DBMS from Ask Computer Systems, has improved this process by supplying a knowledge management module as an add-on. This module allows for the incorporation of rules into applications which use the database (**Jenkins and Grygo 1991**).

The Pseudo-Conceptual Level

The layer of abstraction referred to here as "pseudo-conceptual" is somewhat harder to define. In this work, the pseudo-conceptual layer will refer to a level of integration of knowledge base design and database design in which one particular knowledge representation scheme is modeled via traditional logical database design techniques. At this level, the semantic model becomes of more importance than the syntactic model.

In addition to the physical layer examined above, both (**Anger et. al. 1988**) and (**Rodriguez et. al. 1989**) contain a certain amount of work in the pseudo-conceptual layer. KBS developers can use E-R diagrams to model rule bases in much the same way as databases are modeled. More specifically, their proposal states that "simple assertions of the rule base are viewed as one entity type and the rules as another, with *IF* and *THEN* being relationships between these types." (**Rodriguez et. al. 1989**) Using this method will capture "the information contained within the rules." (**Rodriguez et. al. 1989**)

At Orman's "external level", targeted to end-users of KBSes, rules are depicted graphically (**Orman 1992**). Orman proposes the use of labeled arcs to represent relationships between data items represented by points. Cardinality concepts (e.g.,

SOME, UNIQUE, EACH) are given graphical constructs as well. As in the previous references, though, the graphical representations are limited to applications to rules, thus fitting the definition of the pseudo-conceptual level.

The differences between the physical and the pseudo-conceptual layers cited in the same works are significant. The first set of references to **(Anger et. al. 1988)**, **(Rodriguez et. al. 1989)** and **(Orman 1992)** examined the proposal to take actual rules and store them in a database. In the second set of references to these same three papers, emphasis is placed on taking an existing knowledge base (in all three cases, a rule base) and modeling its semantics via some graphical methodology. Thus, it is the pseudo-conceptual level of database/knowledge base integration at which one can first see an attempt to integrate semantic principles of the two techniques.

The Conceptual Level

At the level of abstraction of KBS-DBS integration referred to as the conceptual level, the particular inferencing technique becomes of secondary importance to the conceptual knowledge schema, in much the same way as the physical database model is of less importance than the logical database schema during the logical design phase of database design. Although previous work has failed to hone in on this level to the extent it has the other two levels, recent literature has seen a trend of research on this level. One example is **(Mattos 1989)**, in which semantic data models and knowledge representation models are characterized as being composed of several abstraction concepts, including classification, generalization, inheritance, element and set association, and element and component aggregation. Mattos further argues that each of these main concepts (classification, generalization, association and aggregation) has inherent reasoning facilities. Additionally, **(Debenham 1992)** presents an argument for building a

"maintainable" knowledge base around Horn clause logic (essentially, a rule-based system) which would, by definition, place his methodology at the pseudo-conceptual level. However, he does defend his approach as being independent of KRS by pointing out that "as long as the knowledge has been modeled rigorously and...this model of the knowledge has been normalized," it "really doesn't matter what language is used to actually implement the knowledge." (**Debenham 1992**) In (**Feldman and Fitzgerald 1985**) the point is made that, while knowledge based systems represent a newer discipline than more traditional information systems, both share common problems in the area of "knowledge representation and acquisition", more than in "technical aspects of programming methods." This common area of concern clearly points to a high level of abstraction in the marriage of the two areas.

Again, the difference between the conceptual level and the pseudo-conceptual level is significant: at this higher level of abstraction, any restriction on inference technique is removed, and the problem becomes one of actually modeling a conceptual base of knowledge with a semantic data model. In (**Borgida 1991**), the point is made that in the database world, more emphasis is placed on "modeling the human conceptualization" of the knowledge domain, while the knowledge base world has just now begun to investigate modeling the conceptual schema vice "modeling the physical storage structures."

In summary, previous examinations of the union of DBSes and KBSes can be separated into three layers of abstraction: physical, pseudo-conceptual and conceptual. The physical layer is the layer at which databases are used to physically manage knowledge, and at which intelligent features are added to DBMSes. The pseudo-conceptual layer begins to examine the use of database design techniques, but generally limits their use to one specific KRM. The most abstract layer, the conceptual layer, suggests the use of database design techniques for any and all KRMs. A combination of

the pseudo-conceptual and conceptual layers will provide the basis for the proposal of this thesis.

Divergence from the KADS Methodology

The KADS methodology has become the most notable KBS design methodology since its origin in 1983 as an ESPIRIT project. Many of the same concerns expressed in this work are also expressed in (**Hickman et. al. 1989**), which is probably the definitive English-language text on the methodology. One such concern is based on the traditional KBS development method, that of rapid prototyping. "(Rapid prototyping) provides very little in the way of support for management issues, which are crucial to successful project control." The authors go on to point to the "deliberate confusion between process and data" as a deficiency in conventional software development methodologies for KBS development. The text claims that entity modeling is not appropriate for KBS development because the process of assigning entities to the real world problem is difficult and the process of assigning attributes to those entities is "very difficult indeed." One item that truly separates the KADS methodology and the other references cited here is that the KADS methodology makes no attempt to integrate knowledge bases and databases, nor does it attempt to separate the knowledge base from the KBS at the logical level.

Knowledge Base Maintenance Using Database Techniques

Additionally, recent research has centered on the area of knowledge base maintenance, and how database design principles can assist. The importance of normalizing knowledge and applying constraints, including the referential integrity constraint, has been discussed in (**Debenham 1992**). (The concepts will be discussed in detail later.) Debenham's work presents three models: the data model, the information

model, and the knowledge model. Basically, the data model is based on the real world realization of the problem, and corresponds roughly to a semantic data model. The information model is analogous to the metadata of a relational database schema, while the knowledge model consists of details about the knowledge representation structure. The data model drives the information model, which in turn drives the knowledge model. Debenham suggests normalization be performed at the data model as it is the easiest to normalize. In addition, non-normalized entities at the data model level can cause a "proliferation" of non-normalized entities at the higher levels. Knowledge base maintenance becomes more manageable with a normalized model, Debenham argues, since all inter-relationships between the component items can be determined more quickly. In a similar manner, Debenham defends applying constraints to the knowledge base (on all three models) as a means to ensure efficiency in the maintenance process.

KNOWLEDGE REPRESENTATION

Just as there are several database models (e.g., relational, network, hierarchical), likewise are there several different knowledge representation models. The most common knowledge representation models are

- Rules
- Frames
- Semantic, or Associative, Networks
- Object Orientation

The inclusion of object orientation as a knowledge representation model could be somewhat debatable; however, when examined at the very basic level, one can see similarities between a frame-representation scheme and an object oriented approach. In addition, object orientation is seen as a means by which intelligence can be added to databases; thus it is included herein as a separate model. Each of these models will be examined in detail to determine what features a modeling tool must provide in order to model their structures.

Rule-based systems are the most commonly known of all KRMs. A rule consists of two parts, a *premise* and a *conclusion*. Rules are generally expressed either as an "IF-THEN" relationship (e.g., **IF** it is August, **THEN** we will have a thunderstorm) or vice versa (We will have a thunderstorm **IF** it is August.) Any number of ANDs, ORs or NOTs can be appended to the premise (**IF** it is August **AND** we are in Central Florida **OR NOT** (I have mowed my yard), **THEN** we will have a thunderstorm.)

A frame-based system collects related knowledge into sets of attribute-value (or slot-filler) pairs called *frames*. The fillers are often subdivided into facets, each of which has its own value (**Gonzalez and Dankel 1993**). Facets may include *range*, *default value*, and *daemons*, procedures which execute upon a pre-defined condition. Frames are ordered in the knowledge base into a hierarchy with IS-A links between the nodes (**Hodgson 1991**). Inheritance plays a major role in frame-based systems as children frames tend to inherit values from parent frames. Using the structure set forth in (**Gonzalez and Dankel 1993**), a frame detailing storm types could be depicted as:

Generic STORM Frame
- Specialization-of: WEATHER
- Generalization-of: (THUNDERSTORM, HAILSTORM, SNOWSTORM)
- Precipitation:
  - Range: (NONE, RAIN, ICE, SLEET, SNOW, HAIL)
  - Default: (RAIN)
- Wind-Speed:
  - Range: (0-150)
- Warning-Type:
  - Range: (NONE, WATCH, WARNING)
  - If-Needed: (WATCH-WEATHER-CHANNEL)
  - If-Modified: (ALERT-MEDIA)
- Lightning-Presence:
  - Range: (NONE, LIGHT, MEDIUM, HEAVY)
  - If-Modified: (CHECK-FOR-THUNDERSTORM)

This example illustrates classification (*Specialization-of* and *Generalization-of*), from which inheritance generally arises, ranges and defaults, and daemons (*If-Modified*, *If-Needed*). The STORM frame will inherit properties of the WEATHER frame, while THUNDERSTORM, HAILSTORM and SNOWSTORM will inherit properties of the STORM frame.

Associative networks, originally termed semantic networks, were developed to represent knowledge in natural language sentences. Their use has grown beyond semantics to encompass physical and causal associations (**Gonzalez and Dankel 1993**).

Associative networks are basically directed graphs whose nodes represent concepts and whose links represent associations between the concepts. These associations can take on many different meanings; classification (instance-of), generalization (is-a) and aggregation (part-of) are three of the more common and important association types (**Mattos 1991**).

Object-orientation (OO) can arguably be presented as a knowledge representation scheme. Its inclusion here is an acknowledgment of the capability of OO to add intelligence to databases. The world of objects has grown to include object-oriented *programming* (OOP), object-oriented *analysis* and *design* (OOA and OOD), and object-oriented *databases management systems* (OODBMS). While each of the three has its own features which are not crucial to this thesis (e.g., the concept of dynamic binding in OOP), all object-oriented approaches share common features, including inheritance, polymorphism and encapsulation. Inheritance in OO is identical to inheritance in frame-based systems. Polymorphism is similar to the concept of generalization. Encapsulation, perhaps the cornerstone of the object world is the process by which data structures and the processes performed upon them (methods) are encapsulated, or combined, into one entity, called a package, class or object type.

In summary, a design methodology for KBSes must meet the requirements of several different knowledge representation schemes. These schemes utilize the following features:

- If-Then relationships between premises and conclusions
- Inheritance
- Generalization/Specialization
- Classification
- Aggregation
- Encapsulation

E-R DIAGRAMS AS A KBS/KBMS DESIGN TOOL

So far, this paper has established the need for a structured design methodology for knowledge-based systems, presented arguments for the integration of knowledge-base and database systems, and examined various knowledge representation models. Building on previous work on the integration of KBSes and DBSes, this chapter will present a design methodology for KBSes which will satisfy the needs of the various KRMs and overcome common problems inherent with KBS design and development. It is the primary intent of this thesis to introduce the use of E-R diagramming as a knowledge base system design tool, and to defend its use by presenting its advantages to various stages of the knowledge base lifecycle.

Semantic modeling has been defined as "the overall activity of attempting to represent meaning." **(Date 1990)** This definition compliments the view of a KRM as a scheme to represent knowledge. It has been argued in **(Borgida 1991)** that semantic data models and KRMs share many similarities, while their differences tend to revolve around the "differing goals to which they subscribe." These similarities include:

- **Object Identity** - both KRMs and semantic models subscribe to the notion that an instance of knowledge or data has its own identity independent of its attribute values or participating relationships.
- **Binary Relationships among Objects** - both support binary (vs. n-ary) relationships among objects (e.g., attributes, slots, properties).
- **Grouping of Individuals into Classes** - Chapter II discussed generalization; the concepts of grouping individuals into classes and generalization are practically identical.

- **Decomposition of Classes into Subclasses** - Chapter II discussed specialization; the concepts of decomposition of classes into subclasses and specialization are practically identical.
- **Constraints** - Both KRMs and semantic models provide means of expressing conditions of validity for attributes.
- **Derived Classes/Relationships** - KRMs and semantic models both have methods defined to control redundant information and enforce its consistency. (**Borgida 1991**)

Drawing upon this list, it is safe to say there is a definite parallel between knowledge representation and semantic modeling. For this reason, this chapter will promote the concept of semantically modeling the knowledge of a KBS as a design aid for KBS development.

Peter Chen's classic entity-relationship modeling and diagramming technique (**Chen 1976**) is arguably the de facto standard for database design in general, and relational database design in particular. As databases have become more intelligent in nature, so too have E-R modeling and diagramming techniques been extended to help developers better keep track of the inherent intelligence of the database. This research will demonstrate how the classic E-R diagram, with extensions, can adequately model the knowledge base of any KBS, regardless of knowledge representation scheme. It will also bring to light some advantages of performing this modeling.

In the decade and a half since Chen presented his very valuable tool, the E-RD methodology has undergone many adaptations. Researchers have proposed extensions to the original model to allow it to model many different types of data and knowledge. The proceedings of the annual Entity-Relationship Approach conferences provide a wealth of

new E-RD extensions. There are object-oriented E-RDs (**Navathe and Pillalamarri 1989**), action-modeling E-RDs (**Feldman and Fitzgerald 1985**), and E-RDs which model both transactional information and conceptual knowledge (**Lazimy 1988**), to name but a few. Elements of many of these "E-RD flavors" will be selected to develop a case for this paper's proposal: entity-relationship modeling and (in particular) diagramming can be used to model the conceptual knowledge base of a knowledge-based system in much the same way as they presently model the logical database of a traditional information system.

Entity-Relationship Basics

A short review of basic E-R modeling reveals three main concepts: *entities*, *attributes* and *relationships*. Peter Chen, who originated both the concept of the entity-relationship model and its graphical partner, the entity-relationship diagram, defines an entity as "a thing which can be distinctly identified." An attribute is a piece of information that describes an entity. Finally, a relationship is defined as "an association among entities." (**Chen 1976**) An example to illustrate these basics is that of a personnel system. The entities of concern are ***EMPLOYEE*** and ***OFFICE***. In this example, employees are assumed to work for one and only one office. The attributes are as follows:

Table 1

EMPLOYEE and OFFICE entities and attributes

EMPLOYEE	**OFFICE**
EMPLOYEE_ID	**ORGANIZATION_CODE**
EMPLOYEE_NAME	**OFFICE_TITLE**
JOB_CLASS_CODE	**MANAGER_ID**
DATE_REPORTED	

Figure 1 shows this example in E-R diagram form.

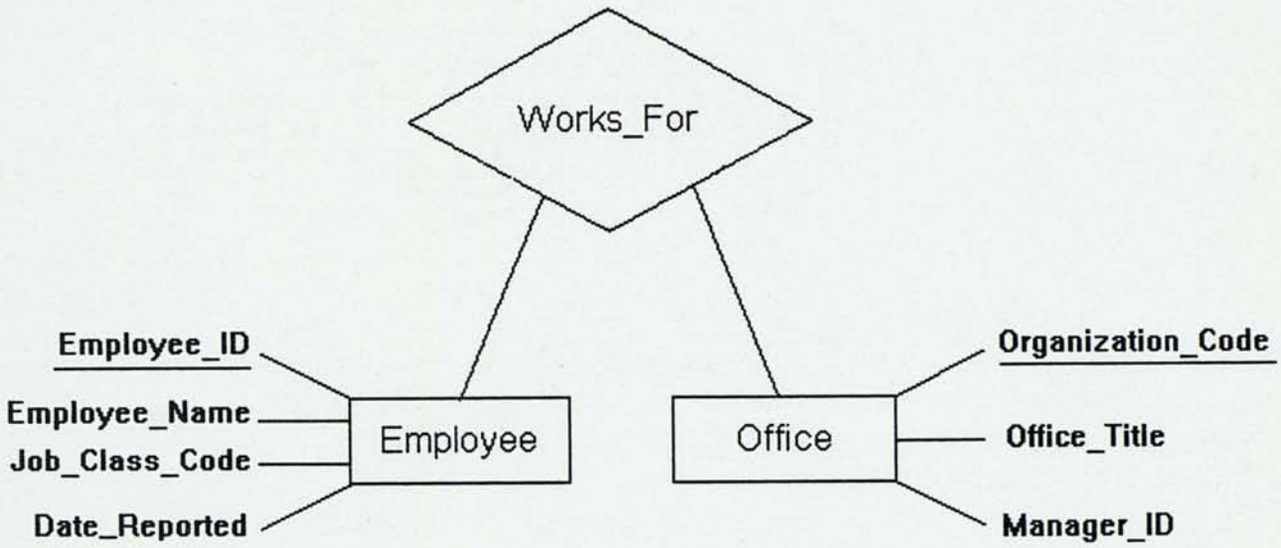

Figure 1 - Sample E-R Diagram

The underlined attributes (**EMPLOYEE_ID**, **ORGANIZATION_CODE**) represent the primary keys of their respective entities. The cardinality of the relationship between the entities is denoted by the "**M**" and the "**1**"; in this example, there is a one-to-many relationship between offices and employees. Although Chen introduced several other features in his essay, these features constitute the bulk of E-RD basics. Appendix A presents a more detailed review of E-R concepts.

Knowledge-modeling Extensions to E-RDs

The first requirement of a knowledge-modeling tool is to provide a model for if-then rules between premises and conclusions. In (**Rodriguez et. al. 1989**), the following diagram is given as an example of how this can be accomplished with standard E-RDs.

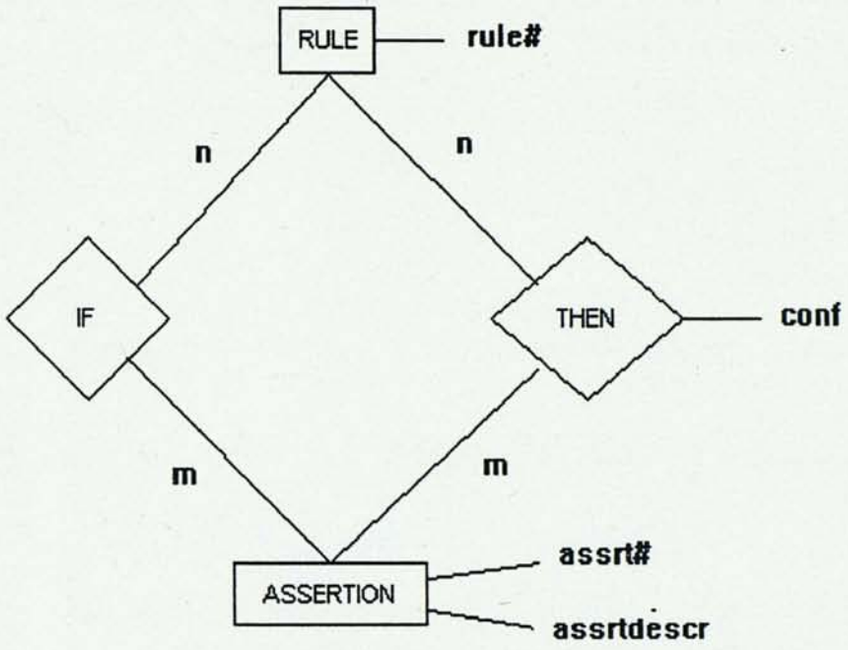

Figure 2 - If-Then E-RD from (Rodriguez et. al. 1989)

This E-RD depicts a many-to-many relationship occurring between the entity **RULE** and the entity **ASSERTION.** This approach differs from more traditional E-R modeling by viewing the rule base as the real world. In traditional database applications, the subset of the real world involved in the problem is modeled as the real world.

A more conceptually-oriented approach to semantically model rules is discussed in (**Feldman and Fitzgerald 1985**). In that work, the use of "action modeling" is presented. They propose this action model to be "constructed in analysis after an entity model has been built," a two-stage approach to knowledge base design similar to the approach espoused in this work. The fact that some sort of behavior modeling must be provided in

order to successfully model a rule excludes the static structure of the entity-relationship model; however, rules do perform their actions on entities, thus some method of depicting them must be provided.

A rule can be considered as an action which occurs as the result of some state of a relationship between one or more entities. As such, a rule should be considered to be an attribute of that relationship. If the rule applies to only one entity, a weak entity and relationship may be created, although this adds an unnecessary step. In this case, the rule may be depicted as an attribute of the given entity. The term "attribute" as used here should not be readily compared to an attribute in a typical database E-RD. Attributes in database E-RDs will become domains, fields or columns in the physical database, while an attribute depicting a rule will see a different mapping in the physical knowledge base. This attribute should be some sort of implementation-independent description of the rule (a "pseudo-rule", comparable to pseudo-code.) The pseudo-rule should either be attached to the relationship as written or identified by a unique identifier and written out elsewhere. This ensures that the relationship between the rule and the entity(ies) the rule references can be determined quickly by visual inspection of the E-RD.

Figure 3 depicts a rule in an E-RD.

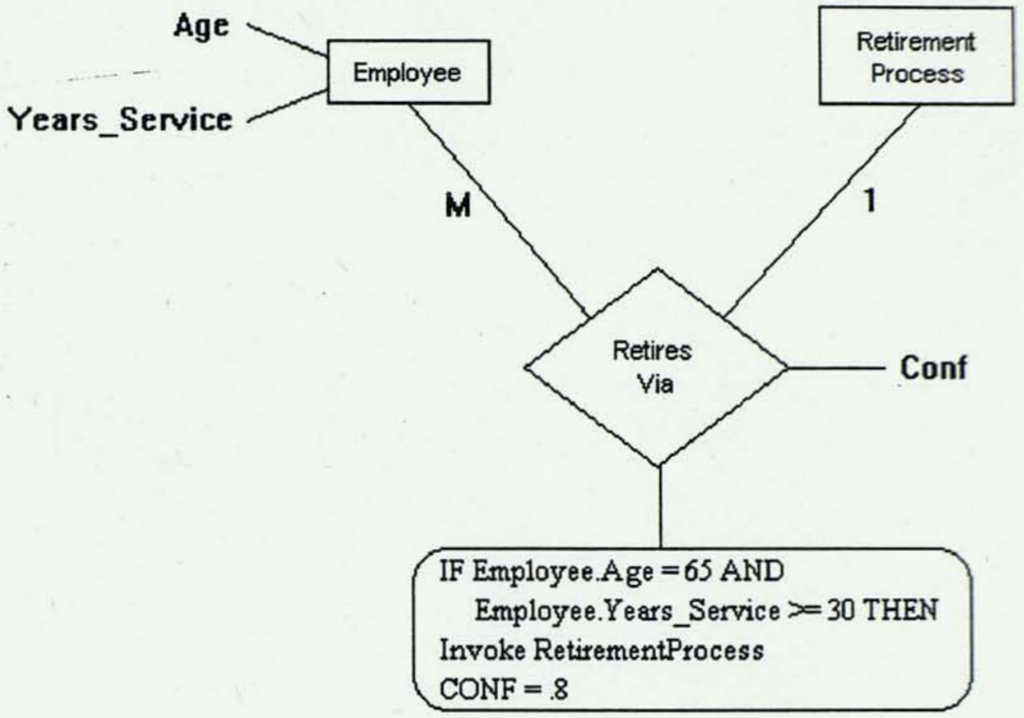

Figure 3 - Depicting If-Then (Rule) relationship in E-RD

The point is made in (**Debenham 1992**) that rules do not always follow the traditional "if-then" format of Figure 2. A semantic model should thus not be limited to if-then relationships simply because the underlying KRM is a rule. However, providing a single diagramming construct to capture all possible rule relationships isn't practical. The method illustrated above allows the designer flexibility in establishing rules.

Classification, generalization, specialization and inheritance all rely on sub- and super-classes. These classes represent a hierarchy from the general (superclass) to the specific (subclass). An entity type which is defined as a superclass will, in an E-RD, be connected to its subclass with a triangle. Multiple subclass entity types each connect to the triangle, which then connects to the superclass entity type. Figure 4 presents an example in which the EMPLOYEE superclass consists of ENGINEER, SECRETARY and SUPERVISOR subclasses.

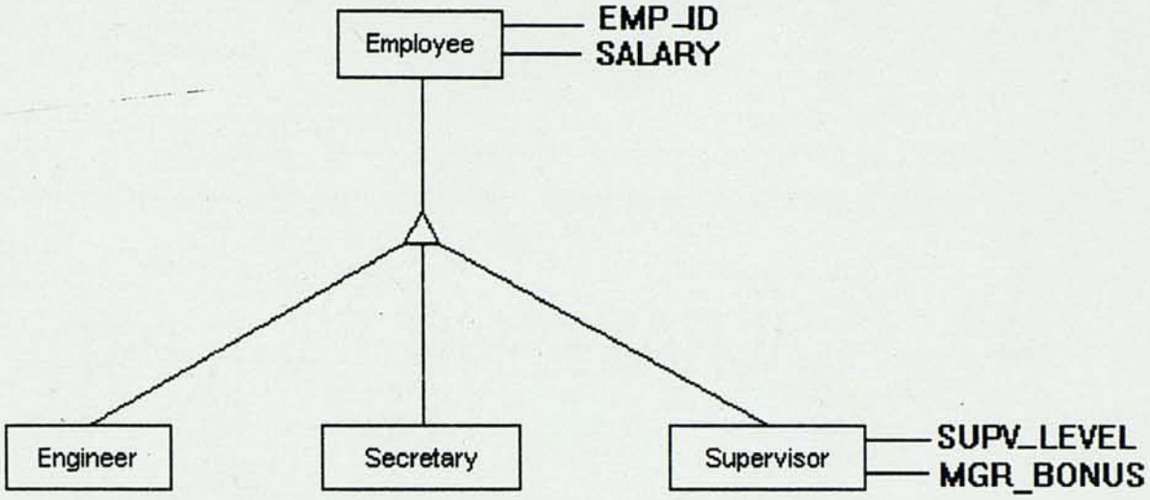

Figure 4 - E-RD illustrating subclasses, superclasses

The presence of a subclass symbol (triangle) represents subclasses; a subclass is assumed to inherit any and all attributes from its parent superclass. Sibling subclasses are not assumed to share additionally defined attributes; if two or more subclass entity types are to share an attribute, that attribute must be explicitly assigned to each entity type. Thus, in Figure 4, all three subclass entity types inherit the attributes EMP_ID and SALARY, while only the SUPERVISOR entity type has SUPV_LEVEL and MGR_BONUS defined.

Generalization and specialization are complimentary concepts, with specialization defined as "the process of defining a *set of subclasses* of an entity type." **(Elmasri and Navathe 1989)** The process of specialization produces subclasses; likewise, generalization produces superclasses. There are several constraints on generalization and specialization which show up in the extended entity-relationship (EER) diagrams defined in **(Elmasri and Navathe 1989)**. These include:

- Predicate definition
- Disjointness