1986

Analyzing a frame-based information system using the relational and entity-relationship data models

Bruce James McTavish
The University of Montana

ANALYZING A FRAME-BASED INFORMATION SYSTEM USING THE RELATIONAL AND ENTITY-RELATIONSHIP DATA MODELS

By

Bruce James McTavish

B. S., Washington State University, 1978

Presented in partial fulfillment of the requirements

for the degree of

MASTER of SCIENCE in COMPUTER SCIENCE

University of Montana

1986

Approved by

Aldon H Wright

Chairman, Board of Examiners

[illegible]

Dean, Graduate School

Date August 29, 1986

McTavish, Bruce J., M.S., August 1986 Computer Science

Analyzing a Frame-Based Information System Using
The Relational and Entity-Relationship Data Models (72 pp.)

Director: Alden H. Wright [illegible]

The importance of modeling the structure of data is increasing as the complexity and size of data bases grows. This importance has created a demand for more expressive and yet easy to understand data models. A data model is an abstract view of a collection of data. This abstract view should provide a clear picture of what the items of interest are in a given application as well as showing how the items are related to one another. The relational model and the entity-relationship model are two data models that will be studied and utilized in this paper. These two models have been used primarily for modeling systems which were implemented in traditional file structures (files which contain records which are made up of fields).

Recently, the author was involved in a project which resulted in an information entry and retrieval system implemented in Lisp. The primary data structure used was the frame. Each individual frame is an entity, and the frame is made up of slots containing information about that entity. This is similar to a record which is made up of fields and which may also represent an entity. Frames are an outgrowth of work done in artificial intelligence (AI). The AI community has its own set of data modeling tools and techniques and these were the techniques used in developing this project.

The goal of this paper is to explore the use of the more traditional data modeling techniques to model an artificial intelligence based implementation construct. In particular, this paper will study the use of the relational and Entity-Relationship data models, to model a frame-based information management system.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Thanks to Dr. Alden Wright. You are an inspiration to your students by being such a curious, eager to learn student yourself.

To Greg Hume, Jim Mitchell, and Mohomad (Saiid) Paryavi, my fellow FIRESYS team members: It has been a very special and rewarding experience working with each one of you. We made a GREAT team!

A very special thank-you to my wife Annie. You have helped me see the value of life, and the benefits of hard work and dedication. You have also put me through school! I love you Annie.

Funding for this project was provided in part through a grant from the Intermountain Fire Sciences Lab, in Missoula, Montana.

Chapter 1

INTRODUCTION

1.1. The Subject Area.

Managing information effectively is becoming more and more important in every working environment. The incorporation of the computer into the business world has changed the means of information management from one of index cards and file cabinets to one of data base management systems and other computer software programs. The need for efficient, useful information entry and retrieval systems is growing dramatically as greater numbers of people want access to ever increasing volumes of data.

The concept of the data base was born to address this need. Martin (Martin, 1976/ p. 4) defines a data base as

> ...a collection of data designed to be used by different programmers...

The intent is to store the data independent of any programs that access it. This step in the evolution of information management made it easier for new applications programs to access the data since the data were stored in a uniform, controlled manner.

This need for information systems has driven the data base designer[1] to

[1]The data base designer is the person(s) who develops and implements the programs that make up a data base management system, or DBMS.

develop tools and techniques for storing and accessing this growing volume of information. The data base designer develops a complete set of programs which access, store, allow the viewing of and provide the security for a data base. This overall set of programs is called a Data Base Management System (DBMS).

A number of models have been developed to provide the data base systems analyst[2] with a logical view of the data to be stored in a data base. This logical view makes it easier to see what facts are being stored and how all of the facts relate to one another. The logical view has no concern for the implementation details of any one DBMS. A name for this overall logical description of a data base is schema. A schema describes all of the types of data that will be stored, and shows the connections or relationships between the data items (Martin, 1976).

The data base systems analyst is not the only person that needs to understand the logical arrangement of the data in the computer. The user must also be able to understand and communicate his or her logical view of the data. This is particularly important for the person representing the user when a new system is first being developed. This person and the systems analyst must be able to express their ideas about the logical data base structure. These ideas will include what information should be stored in the computer and how all of the information is related. A data model will help provide a uniform format to aid in this communication. Data modeling tools are therefore an important link between

---

[2]The data base systems analyst is a data base expert who interacts with the end user of a data base system and makes the decisions about how to utilize the data base in the most productive manner.

the data base systems analyst and the user.

The evolution of ideas and concepts in data modeling has included a process of abstracting further away from the physical implementation of the data base and has aimed more at describing the objects or entities and their relationships. This has allowed the user who is unfamiliar with computer data structures to still communicate easily with a data base systems analyst. The user is able to talk about his domain as he normally does, for example, indicating that part A is related to process B in a certain way. This is in contrast to a user having to understand some implementation concerns such as pointers or indexed files. Whenever there is better communication between individuals, the outcome of a project will be improved.

There have been many models developed to help define how a data base is organized. Some earlier models included CODASYL (Codasyl, 1973), the hierarchical model (Tsichritzis & Lochovsky, 1982), and the network model (Tsichritzis & Lochovsky, 1982). These models are used as a conceptual template in which the data elements and their relationships may be presented. However, these earlier models were closely related to the actual machine representation and this reduced their effectiveness and power as a data model (Martin, 1975). As the models have evolved over the past 25 years they have become easier to conceptualize, easier to modify once a model is constructed, and able to represent many levels of complexity (Martin, 1976). This has made it easier for an untrained end user to sit down with a data base systems analyst and confirm the details of what data go into the data base in what format and with what relationships. There is still a need for even more expressive and more powerful data models to handle the computerization of more and more complicated types of information. (Bic & Gilbert, 1986, Carlson & Arora, 1985). The artificial intelligence community is attempting to use computers for much more sophisticated applications such as natural language processing and expert systems. Improved models are needed to reflect this level of sophistication.

New developments and ideas have had a continuous effect on the field of information management. The artificial intelligence (AI) community is one source of these new concepts. Many AI systems have been developed that store data in frames with slots, rather than the more traditional format of files of records with fields (Fikes & Kehler, 1985). A group of individual frames which store the same type of data may be looked at as similar to a file with a group of individual records. The slots, which when grouped together make up the frame, are similar to the fields which make up a record in a file. Less work has been done in the area of modeling frame-based information systems vs. modeling the traditional file of records representation. Modeling a frame-based system is the area of interest for this paper.

1.2. The Problem To Be Solved.

The intent of this study is to show how two of the more recent modeling techniques can be used to model a frame-based information entry and retrieval system, or data base. The two models are the relational model, as originally presented in (Codd, 1970), and the Entity-Relationship model as originally presented in (Chen, 1976). Recently the author was involved in a project to develop an information entry and retrieval system whose long term objective was to evolve into an expert system. It was felt that a frame-based environment in Lisp would be the most practical and easily modifiable system. This system, code-named FIRESYS, has since been implemented.

The initial design for this system was done via a tree structured hierarchy of the various types of frames (see Appendix B) together with a listing of the frames with their respective slots (see Appendix A. This appears to have done a satisfactory job of organizing the information. There was no formal attempt to utilize any data base tools or techniques as design aids since the original plan was to build an expert system and not a data base. It was felt that an expert system required a different set of development tools than did a data base. A network structure was intentionally avoided during the early design due to its increased complexity over a tree structure. The tree structure together with the other factors involved in the development of the system provided plenty of complexity at the time. It is now felt that by using an established data model to analyze and evaluate this system, the design team and the end users will be able to understand the system more easily and completely. Also, the inclusion of the network complexity into the model will enable the FIRESYS project to more fully implement the users long term needs. It is hoped that the continuing FIRESYS team will use the results from this paper to realize this improvement.

The information system that this thesis will examine was developed between June of 1985 and July of 1986. The work was sponsored by a grant from the

Northern Intermountain Fire Sciences Lab, a division of the USDA. A group of four Computer Science graduate students from the University of Montana, including the author, under the guidance of Dr. Alden Wright, a Computer Science faculty member, was hired to develop a prototype system. The area of interest for this system was fire and its use in forest and on range lands. It was felt that there was a lack of expertise in the area of how fire can be used to improve an area of range or forest land. An expert system seemed to be a solution to this problem.

After several months of interaction between the fire lab personnel and the prototype team it was decided that the fire lab was not ready for an expert system. There was no expert to interact with and it was unclear just what data or knowledge was available to put into an expert system. The decision was made that an information entry and retrieval system was needed. With this type of a system the users could collect and enter the data that was available. As the data is being collected it will become more obvious just what data is available. It will then be easier to construct the expert system. Due to the uncertainty of what data would be entered, a very flexible system that could be easily modified was desired. The concepts of an object-oriented environment and packages were incorporated to facilitate the objective of a flexible system.

1.3. The Framework of This Research.

The system being evaluated, FIRESYS, has already been implemented so this study may be considered a reverse engineering approach to the design of a data base. While one would not want to promote this style of design in most situations, it seems appropriate to the current project for the following reasons. The goal of FIRESYS was to build a prototype information system. To accomplish this the basic specifications for the problem were determined and a working prototype system was developed. The results of this prototype included answers to many of the questions about how the system would actually operate. Another result was the raising of more questions which needed to be addressed. This is where the reverse engineering comes in. The prototype helped to clarify some answers and raise more questions. Once the new questions are answered, it is possible to go back to the beginning and more completely specify the requirements for the system. One of the problems encountered during the system development was the fact that the commissioning personnel at the firelab did not have a clear, consistent idea of what they wanted the system to do. This made it very difficult to obtain a precise specification of the project from which to proceed. In this regard, a prototype was clearly the ideal way to go into this venture. The process of building a prototype forces some questions to be answered during the development of the prototype. Also, more questions are raised as a result of the prototype, and through this process a more complete set of specifications can be established.

In a clearly defined business environment for example, essentially all of the facts are understood and most questions are answered, before any code is written. The process of handling a payroll program is quite exact and the specifications are precise. Payroll is a very well understood domain for computerization. The FIRESYS project was more experimental in nature. Many questions and their answers were not known until the initial prototype system was presented to the users. These new questions can now be dealt with and answers obtained. The changes to the system that are desired due to the new answers are more easily incorporated while the system is still relatively small and more modifiable.

Now that a system does exist it can be evaluated. What was done correctly can be acknowledged and what was done incorrectly can be altered. Thus the prototype development together with reverse engineering is very appropriate for this project. This paper's analysis of the structure of the data base that was built will help FIRESYS grow into a more soundly constructed system.

The goals of this paper are to:

* Model the structure of the FIRESYS data via the relational data model and then via the Entity-Relationship data model.

* Compare these two models with the model that was used for the implementation of FIRESYS.

* Determine if the relational and Entity-Relationship data models are suitable for modeling a frame-based system, such as FIRESYS, and if so, state what improvements they may bring to the FIRESYS project.

The remainder of this paper is outlined as follows:

* Chapter 2 is the development of a relational model of FIRESYS

* Chapter 3 is the construction of an entity-relationship model

* Chapter 4 will present a model of the existing FIRESYS and compare the relational and Entity-Relationship models to this model of the implementation of FIRESYS.

* Chapter 5 is a presentation of suggested modifications to FIRESYS based on the findings of this paper. There are also some concluding remarks on how well the relational and entity-relationship models can

be applied to a frame-based representation of an information management system.

Chapter 2

The Relational Data Model of FIRESYS.

2.1. Background on the Relational Model.

The relational model was first presented formally in (Codd, 1970). Since then many people, including Codd, have expanded on the initial ideas and there is a very strong following for this method of modeling data. This model has made a large step away from the physical machine representation and is a more abstracted, logical view of the data. As Codd put it in his abstract (Codd, 1970/ p. 9)

> Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation).

2.2. The Components of the Relational Model.

2.2.1. The Relation

The primary tool used in the relational model is referred to as a relation. To show the logical structure of a relation an abbreviated format is used. When presenting a relation complete with values a table format is used. The table format is considered a mathematical relation which may be defined as:

$R \subseteq \{[e_1, e_2, \ldots, e_n] \mid e_i \in E_i\}$.

This says that a relation R is a subset of the Cartesian product of its domain sets. The domain sets in the previous example are the set of E_i's. In other words, given the sets $E_1, E_2, \ldots, E_n$ (which do not need to be distinct sets), R is a relation on these sets when it is a set of n-tuples where each tuple's first element e_1 is from E_1, its second element e_2 is from E_2, and so on (Codd, 1970).

One major difference between the mathematical relation and the data base relation is that the data base relation is time varying. Over the course of time, data are added, deleted and modified in the data base relation. Another difference between the mathematical and the data base relation is the ordering of the n-tuples. The ordering of the elements in the n-tuple of a mathematical relation must not be altered. In the relational model this ordering is not a critical factor as long as each member of the n-tuple can be uniquely identified by its attribute name. The attribute names are provided in both the table and the abbreviated formats of the relation. Examples of both of these formats are presented shortly.

2.2.2. Domains, Attributes and Tuples

A domain can be defined as a general set of values from which specific values can be taken. The purpose of the values is to describe some property of an object. For example, from the domain of "integers between 1 and 120" values can be obtained to specify age, speed, or floor-number. From the domain of "character strings of less than 40 characters" values to specify a person's name, the scientific name of a plant, or a habitat-type name can be generated. An attribute is a semantically meaningful named domain, such as age, scientific-name, or habitat-type-name.

When a relation is presented in a table format the attributes are the column headings across the top of the table. Each attribute within any one relation must have a unique name and all entries in that column must be from the domain of the named attribute. A relation in a table format with some actual data values is now presented.

relation name: SPECIES

<u>scientific-name</u>	abbreviation	common-name	...
Sitanion hystrix	SIHY	squirreltail	...
Bromus tectorum	BRTE	cheatgrass	...
Festuca idahoensis	FEID	Idaho fescue	...

The name of the relation is given, the attribute names are at the head of each column, and the primary key column (primary keys are covered later) is underlined. Each row in a table relation is called a tuple. Each tuple is a unique object or entity and the elements of the tuple are descriptive attributes about the object. The values for each attribute are derived from specific domains. The generalization of the individual entities is called the entity type.

In the abbreviated format, the attribute names follow the relation name and are enclosed in parenthesis. An example of the abbreviated format, or the intention of a relation, would be:

SPECIES(<u>scientific-name</u>, abbreviation, common-name, ...)

The name of the relation is SPECIES. The list of attributes includes scientific-name, which is the primary-key for the relation (primary keys will be discussed later), as well as abbreviation, common-name, and others. The primary key attribute name is underlined.

2.2.3. The Primary Key

An important feature in a relation is the primary key. Each tuple within the relation must be uniquely identifiable. This is done via the primary key. The key may be one attribute or it may be a group of attributes. It may even be an artificially generated attribute, strictly for the purpose of being the primary key. The primary key in the SPECIES relation given above is scientific-name. Each species of plant has one scientific name and each scientific name is related to one and only one species of plant. This creates a one-to-one mapping between a species and a scientific name. This way a species can always be uniquely identified by its primary key, the scientific-name.

For a more complete presentation of the formalities of the relational model the reader is directed to (Codd, 1970, Martin, 1975, Martin, 1976, Tsichritzis & Lochovsky, 1982).

2.3. Normalization of Relations.

One very important process in creating a relational model of a data base is normalization (Maier, 1983). The normalization process replaces relationships between data with relationships within a two-dimensional table (Martin, 1975). This table is also called a relation. (see section 2.2.1) For example, a user may specify a set of relationships between data items in the following manner.

* a given species of plant may be found in several habitat-types

* any given habitat-type can be found in only one cover-type

* a given cover-type may be found in several ecosystems

A means of breaking this possibly confusing set of statements into a distinct, clear description is needed. The normalization process helps to achieve this goal. Normalization replaces this seemingly confusing set of connections between data entities with several easy to understand relations. Each relation presents one relationship that needs to be clearly understood. There is a well defined way to join the relations back together temporarily so that the original collection of relationships may be viewed as one group if that is desired.

The result of normalization is a set of relations which provide a user-oriented logical view of the data. This view of the data can be implemented in a variety of ways, and the user does not need to know the method of implementation. This set of relations is known as the logical schema. It is a logical description of the data and the relationships in a data base. A very important advantage of normalized relations is the fact that they can be adapted to changes very easily. As the data base grows and changes over time, new kinds of data may be added to the data base and new views of the data may be developed for new users. Usually, these changes will not affect the existing views nor the existing applications programs that access the data. Even changes in the physical representation may be made without the need to revise the user's view of the data. This feature is very desirable in a data model as it saves lots of money and time in future modifications.

2.3.1. Partial and Transitive Dependencies

There are two concepts that need to be defined in order to appreciate what is happening in the normalization process. These are partial dependency and transitive dependency. Examples will be used to help explain these concepts. Partial dependencies will be presented first.

One of the relations that is used in the FIRESYS project is SEASON-SEVERITY-SPECIFIC-FIRE-EFFECTS, or SSSFE. Let us assume, for the sake of this example, that the relation is as follows.

SSSFE(<u>scientific-name, season, severity</u>,
ave-temp-this-season, fire-effects)

The attribute *ave-temp-this-season* would, by its very meaning, be functionally dependent upon the value of the season attribute. In other words, given a season value, there will be one value that would be the ave-temp-this-season. Season is one of the components of the primary key in the SSSFE relation. Ave-temp-this-season depends upon a part of the primary key value for its value, hence the name partial dependency. This is an undesirable trait in the data base since the same temperature value would be redundantly stored with every tuple that had a particular season as part of the key. Aside from the storage considerations of redundancy, if the value of ave-temp-this-season needed to be changed, it must be changed in every place it was stored. This is the problem of consistency of data. The following example illustrates this problem.

relation name: SSSFE

scientific-name	season	severity	ave-temp-this-season	fire-effects
cheatgrass	spring	mild	67	killed
cheatgrass	summer	hot	87	killed
wheatgrass	spring	mild	67	damaged
wheatgrass	spring	hot	67	killed

Let us assume that a new study was done and it was determined that the ave-temp-this-season for spring should actually be two degrees higher than the current value. All occurrences of that value wherever they occurred in the relation would need to be changed. A better solution, and one which would remove the partial dependency, would be to create a new relation called SEASON-AVERAGE-TEMP. This relation would store a list of seasons together with the average temperature for that season. The season attribute would then be in both the SSSFE and the SEASON-AVERAGE-TEMP relations while the ave-temp-this-season attribute would be in only the SEASON-AVERAGE-TEMP relation. The new relations would be as follows.

SSSFE(<u>scientific-name, season, severity</u>, fire-effect)
SEASON-AVERAGE-TEMP(<u>season</u>, ave-temp-this-season)

and the tables would look like this.

relation name: SSSFE

<u>scientific-name</u>	<u>season</u>	<u>severity</u>	fire-effects
cheatgrass	spring	mild	killed
cheatgrass	summer	hot	killed
wheatgrass	spring	mild	damaged
wheatgrass	spring	hot	killed

relation name: SEASON-AVERAGE-TEMP

<u>season</u>	season-ave-temp
winter	22
spring	67
summer	87
fall	56

When the value for ave-temp-this-season for spring needed to be changed there would be one change made in the data base and everything else would be up to date.

As a reminder to the reader, there is no ave-temp-this-season attribute in the actual SSSFE relation for FIRESYS. Also, an important note here is that in order for there to be a partial dependency the primary key of the relation must be a multiple key. That is, there must be more than one attribute in the key in order for some non-key attribute to be partially dependent upon the key of the relation.

Transitive dependency is the other concept to be discussed. Let us again set up a hypothetical relation to satisfy the needs of our example. Assume the following relation exists.

SPECIES(<u>scientific-name</u>, flower-color, pollinating-insect)

Let us also assume the following: the value of scientific-name, the primary key in SPECIES, determines the value of flower-color; flower-color, a non-key attribute, determines the value of pollinating-insect. There is now a non-key attribute whose value is dependent upon another non-key attribute. Pollinating-insect is dependent upon flower-color. This situation is similar to that of partial dependency, but now neither of the attributes is a part of the primary key. The following table clearly shows the redundancy involved in a transitive dependency.

relation name: SPECIES

<u>scientific-name</u>	flower-color	pollinating-insect
rhodeii dendroni	red	red-bellied-bee
azaleaii plantii	yellow	yellow-bellied-fly
rosei prettyi	red	red-bellied-bee
carnationi yellowi	yellow	yellow-bellied-fly

The removal of the transitive dependency is accomplished by creating a new relation. The new relation would be flowercolor-pollinatinginsect. It would contain a list of colors together with the insect that pollinates that color of flower (this is a contrived relationship between color and insects). The flower-color attribute would then be in both relations and the pollinating-insect attribute would be only in the colorofflower-pollinatinginsect relation, as shown below.

relation name: species

scientific-name	flower-color
rhodeii dendroni	red
azaleaii plantii	yellow
rosei prettyi	red
carnationi yellowi	yellow

relation name: flowercolor-pollinatinginsect

flower-color	pollinating-insect
yellow	yellow-bellied-fly
red	red-bellied-bee

2.3.2. The Three Normal Forms

There are three levels of normalization that are applied to relations. They are first normal form, second normal form and third normal form.

2.3.2.1. First Normal Form

Achieving first normal form involves setting up a table with all of the desired attributes for an entity type across the top of the table. These become the headings for the columns. Next, the data is input as tuples, and these make up the rows in the table. This table must meet the following five properties, in order for it to be in first normal form. (Martin, 1976):

1. Each entry in a table represents one data-item; there are no repeating groups.

2. They are column-homogeneous; that is, in any column all values are derived from the same domain.

3. Each column is assigned a distinct name; a unique attribute name

4. All rows are distinct; duplicate rows are not allowed. the primary key helps insure uniqueness.

5. The ordering of the rows and columns can be changed without affecting either the information content or the semantics of the data. the columns must be column-homogeneous and the rows must be distinct, but the ordering of both is insignificant.

The first property listed requires some additional discussion as it raises the following question. When is something a repeating group and when is it simply a group of values? The problem involves an attribute that contains a list of values. This situation occurs several times in the FIRESYS data. For example, within one species there may be a list of common-names. A table representation of this example would like like this.

relation name: SPECIES

<u>scientific-name</u>	common-name	abbreviation	color ...
Sitanion hystrix	squirreltail rabbittail birdtail	SIHY	green ...
Bromus tectorum	cheatgrass stealgrass	BRTE	tan ...
Festuca idahoensis	Idaho fescue	FEID	brown ...

A list of values is not allowed in first normal form which means that this table is not in first normal form. There is a list of common names for two of the species in the table. There are two ways of handling an attribute which has a list of values.

* Treat the list as one item, in which case the attribute can remain a non-key attribute of the relation. In respect to our example of a species with a list of common-names, there would still be only one tuple for a given species.

* Treat each component of the list as an individual item, in which case it becomes a part of the primary key. From our example, this would cause a new tuple to be created for each common-name stored.

The results of the first method would be a relation just like the one in the previous example except that the common-name attribute should probably be renamed list-of-common-names. These common name values are now not suitable to use as a means of identifying or locating these tuples in the example relation or any other tuples in any other relations. The value for the list-of-common-names attribute should be thought of as the totality of the list, as opposed to a list of distinct values.

The second method given above involves creating a new tuple for each common name value in the list. The result is an additional relation as shown by the following example. Note that the SPECIES relation still exists, but does not contain any common name values. The new relation now contains the common-name attribute.

relation name: SPECIES

<u>scientific-name</u>	abbreviation	color ...
Sitanion hystrix	SIHY	green ...
Bromus tectorum	BRTE	tan ...
Festuca idahoensis	FEID	brown ...

relation name: SPECIES-COMMON

scientific-name	common-name
Sitanion hystrix	squirreltail
Sitanion hystrix	rabbittail
Sitanion hystrix	birdtail
Bromus tectorum	cheatgrass
Bromus tectorum	stealgrass
Festuca idahoensis	Idaho fescue

How to handle this problem can be a difficult decision. The main factor in this decision should be how the user envisions the items in the list being used. If the items in the list will be used as a means of identifying any tuple in any relation, then the list should not be kept as one item. Instead, a new relation should be established and each item in the list is a component of one tuple. If the items in the list are strictly data values that are related to an entity, and they will not be used as a means of identifying that entity, then it is probably acceptable to leave the items in a list.

Another factor in the decision of how to handle a list of values concerns the possibility of other attributes that might be associated with the values in the list. If new attributes will be associated with the list of values, then the second method should be employed. It will be relatively easy to add any new attributes to the new relation with each list item in its own tuple. In contrast, it would be much more difficult to incorporate any newly desired attributes and associate them with individual elements of a list, if the first method were used and the items were all in one list.

One concern which is at the implementation level involves the attribute field length. Most data base implementations require a fixed length field to be specified for each attribute. In determining this size, the maximum length of a value should be used, within reason. When an attribute is made up of a list of items, it may be difficult to determine how many items to allow for. Also, once the maximum length is determined, can that much storage space be afforded for this attribute? The storage space may also be a factor in the decision of how to handle a list of items.

2.3.2.2. Second & Third Normal Form

Second normal form is obtained when a relation is in first normal form and there are no partial dependencies of non-key attributes on primary key attributes. (see section 2.3.1 for a presentation of partial dependencies.)

Third normal form is achieved when a relation is in second normal form and there are no transitive dependencies of non-key attributes on primary key attributes. (see section 2.3.1 for a presentation of transitive dependencies) A data base in third normal form will be minimally redundant and will avoid update anomalies. Update anomalies are the result of additions, deletions, or modifications to the data base which leave inconsistencies or conflicting values. It is very desirable to avoid update anomalies in a data base operation.

A full detailed description of the normalization process will not be presented in this paper. The relational model of FIRESYS will be given, and the third normal form properties will be described.

2.4. The Data To Be Modeled

A prototype system has already been implemented for FIRESYS. Through this development a fairly well defined list of data items, together with the relationships between the data, has been generated. For a full listing of these data items and their relationships the reader is directed to Appendix A.

There are five major entity types of interest. There are other entity types whose importance to the overall structure of the FIRESYS data is less important. A brief view of these other entity types, and how they relate to the five major entity types, will be presented in section 2.4.3. The primary entity types are:

* Ecosystems
* Cover-types
* Habitat-types
* Species
* Season-Severity-Specific Fire Effects

2.4.1. The Entity Relations.

A relation is created for each of the objects or entity types of importance to FIRESYS. A list of attributes is associated with each object. From this list, a primary key is selected. Each of the relations is presented in third normal form, and this fact will be detailed for each relation. This presentation of the data assumes that for any attribute containing a list of values the entire list is treated as a single value. (see section 2.3.2.1 for a discussion of a list of values in an attribute.)

Table 2-1 shows the relations with the attributes of interest for the five primary objects. The primary key is the underlined attribute. Only a few of the actual attributes for these relations are shown in order to keep the presentation simple.

Table 2-1: Relations for Primary Objects in the FIRESYS Model

ECOSYSTEM(ecosystem-name, classification-key,
kuechler-vegetation-types, ...)

COVER-TYPES(cover-type-name, site-characteristics,
vegetative-composition, ...)

HABITAT-TYPES(habitat-type-name, distribution,
successional-trends, ...)

SPECIES(scientific-name, life-form, abbreviation, ...)

SEASON-SEVERITY-SPECIFIC-FIRE-EFFECTS(
season, severity, scientific-name,
effect, certainty-factor, ...)

The relations in Table 2-1 are in third normal form. The following facts support this claim. All values of each attribute in each relation are fully dependent upon the entire primary key of that relation. For example, in the COVER-TYPES relation with the key cover-type-name, all other attributes, some of which are not shown, depend entirely upon the value of cover-type-name. There are no partial dependencies and there are no transitive dependencies. In fact, there could not be any partial dependencies since the primary key is a single attribute value.

The ECOSYSTEM, HABITAT-TYPES and SPECIES relations also have single attribute primary keys. The values for all of the attributes in these three relations depend entirely upon the value of their respective key. Due to their having only a single attribute key, none of these relations has any partial dependencies. Since the values for all of the remaining attributes is determined strictly by the value of the respective primary key, there are no transitive dependencies. Based on these factors, the ECOSYSTEM, HABITAT-TYPES, and SPECIES relations are also in third normal form.

The primary key for the SEASON-SEVERITY-SPECIFIC-FIRE-EFFECTS relation is made up of three attributes. All of the remaining non-key attributes are fully dependent upon the combined values of the three part primary key. In other words, once the three primary key attribute values are determined, there is only one possible value for each of the remaining attributes. Therefore there are no transitive dependencies. None of the non-key attribute values can be determined until all three primary key values have been established. This means that there are no partial dependencies. This shows that the SSSFE relation is in third normal form.

2.4.2. The Relationship Relations.

Another set of relations is required in order to represent some of the relationships that the user is interested in. A separate relation is needed to represent the following two facts.

1. a cover-type may exist in more than one ecosystem
2. an ecosystem may contain more than one cover-type

This is an example of a many-to-many relationship between ecosystems and cover-types. This same type of many-to-many relationship needs to be shown between habitat-types and species. The relations for these relationships are shown in table 2-2.

Table 2-2: Relationship Relations in the FIRESYS Model.

ECOSYSTEMS-COVERTYPES(<u>ecosystem-name, cover-type-name</u>)

HABITATTYPES-SPECIES(<u>habitat-type-name, scientific-name</u>,
species-percent-cover-in-hab, fire-effects, ...)

The purpose of the first relation is the following. Given an ecosystem-name, find all the cover-types that exist in that ecosystem. First, all tuples in the ECOSYSTEMS-COVERTYPES relation with the desired ecosystem-name are located. Then the cover-type-name attribute can be read from each of these tuples. This provides a list of cover-types that exist in a given ecosystem. With the same relation it is possible to determine in which ecosystems a given cover-type might be found. The first step is to locate in the ECOSYSTEMS-COVERTYPES relation all tuples with the desired cover-type-name. The list of ecosystem-name attributes associated with the selected cover-type-names can then be read.

The same two types of searches may be done with the HABITATTYPE-SPECIES relation. Other information is provided in the HABITATTYPE-SPECIES relation. The species-percent-cover-in-hab attribute is an attribute of the relationship between the species and habitat-type entity types. It is not an attribute of either of the two individual entity-types that the relation is dealing with.

If there is information desired about the species that exist in a particular habitat-type, it can be found in the following manner. First the habitat-type entries in the HABITATTYPE-SPECIES relation are located based on the habitat-type-name. Then the scientific-name attribute associated with each habitat-type is read. Each scientific-name can then be looked up in the SPECIES relation, and the desired information on the species can be examined.

These relations in Table 2-2 are also in third normal form. In the case of the ECOSYSTEMS-COVERTYPES relation there are only primary key attributes. This precludes any chance of there being either partial or transitive dependencies. In the HABITATTYPE-SPECIES relation, the non-key attributes shown are fully dependent upon both elements of the primary key for their value. This means that there are no partial or transitive dependencies.

An additional relationship relation will be presented that deals with the problem brought up in section 2.3.2.1. That problem involved a list of values for one attribute. In the original presentation of the entity relations in section 2.4.1, the assumption was made that all lists of values for a single attribute would be treated as a single item. The other means of handling a list of items is to separate the items in the list and create new tuples for each item. (see section 2.3.2.1). The relations that are a result of this other method will be presented now.

A list of common-names for a given species needs to be represented. Common name is an attribute that may be used in order to locate a particular species tuple. The elements in the list of common-names will be separated and additional tuples will be created in the first normal form table. Through the normalization process this eventually creates another relation. The resulting relation would look like this.

SPECIES-COMMON(<u>species-name, common-name</u>)

This relation will determine the values for the common-names associated with a given species-name. This relation will also provide the species-name when given a common-name. More than one species name may be known by the same common-name and a species may have more than one common-name. This is why the primary key is made up of both attributes. Due to the fact that both attributes are part of the primary key, there is no chance for partial or transitive dependency. Hence, this relation is also in third normal form. This type of a relation is a common result of an initial list of values being separated into additional tuples.

2.4.3. Additional Relations.

Through the development of the prototype it was observed that the five primary entity types had a large volume of information stored with them. For example, there were as many as forty attributes to be associated with the species entity type. In order to provide the user with a more convenient organization of the data these forty or so attributes were broken into a group of entity types of their own. A new relation was created for each of these new entity-types. This partitioning of the data was not based on the needs or requirements of the relational model nor on the normalization process. It was done for the sake of simplifying the organization of the data into smaller conceptual blocks which are easier for the user to deal with. These additional relations are being presented separately due to their lack of importance to the overall data model of FIRESYS from the relational point of view.

There are five relations that are directly related to the SPECIES relation. All five have as their primary key, scientific-name. They may be considered an extension of the SPECIES relation. They are in third normal form, as all of the attributes of each relation are fully functionally dependent upon the scientific-name primary key. The five relations are presented in Table 2-3.

Table 2-3: Additional Relations Relating to SPECIES

DISTRIBUTION-AND-OCCURRENCE(<u>scientific-name</u>,
BLM-physiographic-region, SAF-cover-type, ...)

VALUE-AND-USE(<u>scientific-name</u>, palatability, cover-value, ...)

BOTANICAL-AND-ECOLOGICAL-CHARACTERISTICS(<u>scientific-name</u>,
growth-form, raunkiaer-life-form, ...)

FIRE-ADAPTIVE-TRAITS-AND-SURVIVAL-STRATEGIES(<u>scientific-name</u>,
lyon-stickney-fire-survival-strategy,
rowe-mode-of-persistence, ...)

FIRE-EFFECTS(<u>scientific-name</u>, fire-effect-on-plant,
plant-response-to-fire, ...)

There are an additional two relations that apply to the HABITAT-TYPE relation just as the previous five relations applied to the SPECIES relation. These two are given in Table 2-4.

Table 2-4: Additional Relations Relating to HABITAT-TYPE

HABITAT-MANAGEMENT-CONSIDERATIONS(habitat-type-name,
livestock-range, wildlife-habitat, ...)

HABITAT-FIRE-ECOLOGY-AND-EFFECTS(habitat-type-name,
immediate-fire-effects-on-community,
long-term-community-response-to-fire, ...)

2.5. Summary for the Relational Model

The relational model has proved itself to be more powerful and complete than the simple tree-structured model that was used for FIRESYS. (see section 4-1 for the model of FIRESYS). It is a relatively straightforward process to establish the relations and normalize them. The structure of the data has been made very clear by using a logical well defined model. The users presented the data in a very unstructured arrangement and, through the use of the relational model, the data became organized into a precise unambiguous structure. This shows that there are benefits of using a well organized data model such as the relational model. It forces a clear picture of the data to be drawn, including what data items are involved and what relationships exist between various data items. The model is created without any of the complexity of the access paths or the implementation process. This allows all of the concentration and study to go to the data structure alone. This is an important separation of activities in the development of a data base system.

Chapter 3

The Entity-Relationship Data Model.

3.1. Background on the Entity-Relationship Model.

The Entity-Relationship, or E-R, model was presented primarily in (Chen, 1976). The ideas of entities and relationships have been dealt with before, but Chen presented the entire model as a well thought out concept. One of the motivating factors for Chen's work was the desire to represent more semantic information along with a list of data items and their relationships. Some interesting semantics of data would include the following.

* two data items are related, but more than that, one of them depends upon the other to justify its existence

* again, two data items are related, but one of them can be identified, only through the identification of another item

An example of the first case would be that a certain species of plant depends upon the existence of some habitat-type in order for the species to be a valid entry in the FIRESYS data base. For the second case, the SSSFE entities are not uniquely identified until the species name to which it is related has been provided. These are facts about the data that the user is interested in and it is desirable for a data base to be able to know and represent these facts.

There is much support for the inclusion of relationships, as well as entities, as distinct components of a data model. An analogy is presented in (Hartzband &

Maryanski, 1985) that equates the tables of the relational model to nouns in a language, in terms of their expressive power. Hartzband and Maryanski then state that the addition of relationships to the data model is similar to the addition of verbs to a language. It creates a much more descriptive capability in the data base model. Chen, in (Chen, 1976), makes the claim that the separation of entities and relationships in the data model makes it easier to identify functional dependencies among data items. This helps to provide a better understanding of the true relationships between data items. Determining functional dependencies also aids in achieving the equivalence of the relational model's third normal form.

3.2. The Components of the Entity-Relationship Model.

3.2.1. Entities, Entity-sets, Relationships, & Relationship-sets

The primary components of the E-R model are entities and relationships. Chen, in (Chen, 1976/ p. 10), makes a very simplistic definition of them both.

> An *entity* is a "thing" which can be distinctly identified.
> A *relationship* is an association among entities.

Examples of entities would include a specific person, company, event or species of plant. Examples of relationships would include father-son, department-manager or habitattype-species.

Entities are members of entity-sets on the basis of a test predicate. Peter Ng, in (Ng, 1981/ p. 86), defines an entity-set in the following way:

> Let e denote an entity, which is an object that can be distinctly identified. An *entity-set* E is defined as $E = \{e|p(e)\}$, where p is the aforementioned test predicate.

Entity-sets do not need to be mutually disjoint. That is, a member of one entity-set may be a member of another entity-set. For example, a specific person may be a member of the entity-set MALE and also a member of the entity-set PERSON. Entity-sets are the logical grouping of a set of entities.

Relationships are members of relationship-sets. A relationship-set is a mathematical relation among n entities which are themselves members of entity-sets. The mathematical definition as presented in (Tsichritzis & Lochovsky, 1982/ p. 177-178) is:

If RS is a relationship-set, it can be defined as:

$$RS \subseteq \{[e_1, e_2, ..., e_n] \mid e_i \in E_i\}$$

where e_i is an entity that is a member of the entity-set E_i.

It is important to note that $[e_1, e_2,...,e_n]$ is an ordered tuple and also a relationship. The individual relationship is a member of the relationship-set.

Entities, entity-sets, relationships and relationship-sets will be presented in the following examples. Sitanion hystrix is the name of a species of plant and as such, it is an entity. The collection of all species would constitute an entity-set. A possible predicate test for this entity-set could be that "x is a species if x is listed in the FIRESYS computer files". Another example of an entity might be a habitat-type named Artemesia arbuscula/Poa sandbergii (abbreviated to AAPS). The collection of all habitat-types would make another entity-set. A possible predicate test for this set might be that "x is a habitat-type if x is in the FIRESYS computer files".

A relationship exists between AAPS and Sitanion hystrix, since the species

Sitanion hystrix is found growing in the AAPS habitat-type. A relationship-set exists between the entity-set made up of species and the entity-set made up of habitat-types. The relationship-set is a subset of the Cartesian product of these two entity-sets. That is, it would be a set of species - habitat-type pairs such that the species did grow in the habitat-type that it was paired with. The pair [Sitanion hystrix, AAPS] would be a relationship which is an element of the species-habitat-type relationship-set.

One problem that must be dealt with in the E-R model is the determination of whether something is an entity or a relationship (Bic & Gilbert, 1986). For example, is a marriage a relationship between two entities of type person, or is marriage an entity with attributes of husband and wife. It really depends upon the intended use of the data base and the decision is up to the data base designer. It is a subjective decision and one that can haunt the data base designer if it is made incorrectly.

3.2.2. Roles, Attributes, & Value-sets

The concept of a role can eliminate the need for a tuple to be an ordered list of entities. A role is the purpose or function that an entity serves in a relationship. For example, in a species-habitat-type relationship two roles can be identified. They are individual-plant and plant-grouping. Many times the role played will have the same name as the entity itself. A role is different than an attribute of an entity or an attribute of a relationship. A role is the function that an entity plays in a relationship.

Entities and relationships do have attributes that may be thought of as the descriptive components or information relating directly to the entity or relationship. The values that an attribute brings into an entity-set or relationship-set come from a value-set. A value-set serves basically the same purpose as the domain in the relational model. In the E-R model, an attribute is a function which maps from an entity-set or relationship-set, to a value-set or the Cartesian product of value-sets. Chen describes it formally in (Chen, 1976 / p.12) as:

$$f: E_i \text{ or } R_i \rightarrow V_i \text{ or } V_{i1} \times V_{i2} \times \ldots \times V_{in}$$

Constraints may be placed on the values allowed in a value-set. For example, a value-set may be defined as "the set of BLM Physiographic Regions", which would constrain the values to that set of region names that the BLM (Bureau of Land Management) has set forth.

The number of items allowed in a relationship is another factor that is presented in the E-R model. A relationship may be one-to-one, one-to-many or many-to-many. This information is given explicitly, and is another way in which the E-R model gives more of the semantics of the enterprise being modeled.

3.2.3. Existence and Identity Dependencies

Two semantically helpful features that can be expressed in the E-R model are the existence dependency and the identification dependency. The existence dependency deals with the fact that sometimes one piece of data in a data base is valid only if another piece of data exist. An example would be that the existence of the species entities depends upon the existence of an associated habitat-type. If all of the habitat-types in which a given species are found are eliminated from the data-base, then the given species must also be eliminated. If the habitat to species relationship was a one-to-many relationship, then if the one habitat-type that a species grew in were eliminated, then the species would also need to be eliminated. Dealing with this concept explicitly in the data model helps to insure that the data-base correctly represents the real world as much as possible. The dependent entity-set, in this case the species entity-set, is termed a weak entity-set and the relationship involved is termed a weak relationship-set.

The other dependency, the identification dependency, is another real world fact whose semantics can be shown in the E-R model. Life is full of entities where the means of identifying them is by saying that they are related to some other entity. For example, in FIRESYS, there are a great number of Season-Severity-Specific-Fire-Effects (SSSFE), but in order to give any of them any valid meaning they need to be related to a specific species. The SSSFE entities are identified by associating them, or relating them, with a species.

In any case where there is an identity dependence there is also an existence dependence. In this case, this means that if a given species is deleted from the data base, then the related SSSFE's must also be deleted. A lone SSSFE is a meaningless, unidentifiable entity without its species. Due to the fact that an identity dependence implies an existence dependence, anytime there exist an identification dependence there exist a weak entity-set and a weak relationship-set.

There can be an existence dependence without an identity dependence. For example, the species entity-set is dependent upon the habitat-type-set for its existence, but any species can be uniquely identified by its own species-name.

In contrast to the weak entity-sets and weak relationship-sets there exist regular entity-sets and regular relationship-sets. When an entity is not dependent upon another entity for its existence, the entity-set is called a regular entity-set. When the entities that are involved in a relationship-set are all regular entities, the relationship-set is termed a regular relationship-set.

Listing these types all at one, there are regular and weak entity-sets and regular and weak relationship-sets. The ecosystem entity-set is the only regular entity-set in FIRESYS. All of the other entity-sets have an existence dependency and so are all weak entity-sets. All of the relationship-sets involve at least one weak entity-set and therefore they are all weak relationship-sets.

3.2.4. Primary Keys

One more concept in the E-R model is that of the primary key. As was the case in the relational model the primary key in the E-R model is a unique means of identifying an individual item out of a group of items. In an entity-set it would be the means of selecting a specific entity from an entity-set. For example, in the SPECIES entity-set, a specific species of plant can be uniquely identified by using the species-name. The species-name is the primary key and will always be a unique string of characters for each species. In a relationship-set, the primary key is made up of the primary key of each of the entity-sets that are involved in the relationship.

It is not a requirement that every entity-set have a primary key in the E-R model. In the case of an identity dependent entity-set there is no means of uniquely identifying any of the entities without the use of a relationship with another entity-set. The dependent entities do not have a primary key. Once the relationship is established the dependent entities are able to be uniquely identified, though unique identification is not a requirement of the identity dependent entity-set itself.

3.3. The Entity-Relationship Diagram

The means of presenting the E-R model is primarily through the Entity-Relationship Diagram, or ERD. Most of the concepts that are involved in the ERD have been presented. The means of diagraming these concepts will now be given.

Entity-sets are pictured as labeled rectangles in the ERD. Relationship-sets are shown as labeled diamond shapes. These two objects are connected by arcs to show which entity-sets are involved in which relationship-sets. Figure 3-1 gives a simple example of these ideas. It involves the species and habitat-type entity-sets which are related by the habitat-species relationship-set. Note the letters next to the arcs. These letters indicate that this is a many-to-many relationship. This tells us that a given habitat-type may contain many species and also that a given species may be a member of many habitat-types.

Another important fact is that the arc from the habitat-species relationship-set to the species entity-set is a directed arc. Also, there is an E in the relationship-set diamond, and the species box is a double box. This is how the existence dependency is denoted in the ERD.

To show that one entity depends upon another entity for its identification a

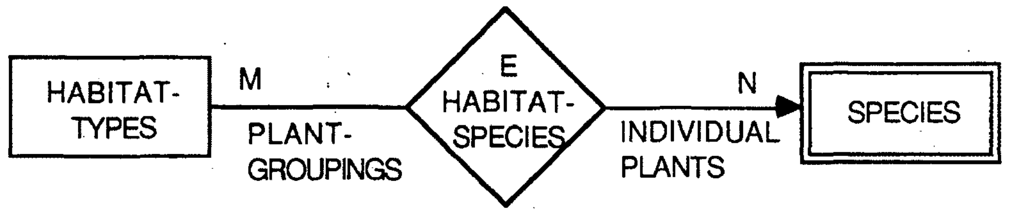


Figure 3-1: The Basic Entity-Relationship Diagram.

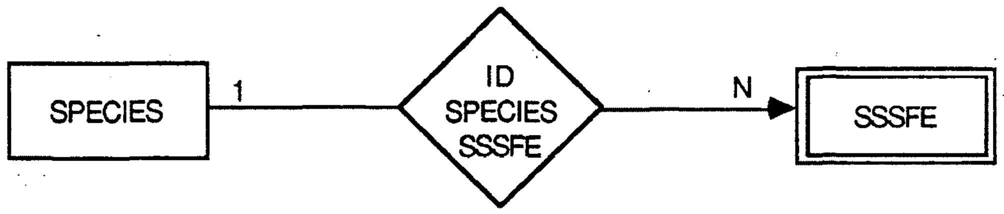


figure 3-2: An Identification Dependency in the ERD.

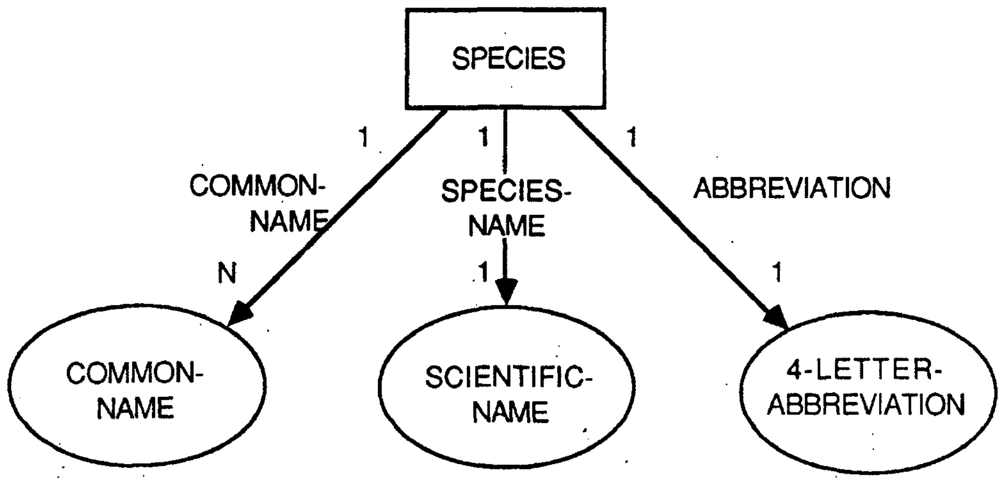


Figure 3-3: Attributes in the ERD.

similar notation is used, except that an ID, rather than an E, is placed in the relationship between the entity-sets. An example of this identity dependence is shown in figure 3-2.

Roles are presented in the ERD by labelling the arc between an entity-set and the relationship-set. An example of this can be seen in figure 3-1. The habitat-type is shown to be serving the plant-grouping function or role while the species is serving the individual-plant role. To someone looking at this ERD who does not know what a habitat-type is, these role names give more semantic meaning and may help the reader to understand what the purpose of the relationship is.

Attributes are shown on the ERD by circles, as in figure 3-3. The attribute name is shown next to the connecting arc, while the value-set name is given inside the circle. The two names may be the same but when the value-set shows some constraint or general quality of the attribute, it will use a different name. The mapping between entity-sets or relationship-sets and their corresponding value-sets can also be shown. The example in figure 3-3 shows that one species may have multiple common-names, while one species will only have one scientific-name and only one four letter abbreviated name.

It can become very messy to attempt to show all the entity-sets, relationship-sets and attributes for one schema in one figure. The attributes are often given in separate figures. The entity-sets and relationship-sets, together with any mapping values, roles, identity constraints and existence constraints are often enough to fill any one ERD. For a complete ERD of FIRESYS the reader is directed to Appendix C.

3.4. Normalization as Applied to the E-R Model.

The normalization concept of the relational model was presented in chapter two. It was mentioned that this was a very important process in the relational model. The results of normalization, and in particular of third normal form, can also be achieved via the E-R model (Ng, 1981 / p. 92b). The benefits include minimal redundancy and freedom from update anomalies brought on by changes in the data base. The method for achieving the equivalence of third normal form is presented in (Ng, 1981 / pp.92-96). The same partial and transitive dependencies that were described in section 2.3.1 are used to analyze the entities and their attributes as well as relationships and their attributes. The verbal description of the data base application as presented by the user is referred to in order to insure that all of the semantics are dealt with properly. Additional analysis techniques are also used and these include a heuristic approach (Ng, 1981 / p. 96). This normalization process was not used for this paper since the groupings of attributes into entity-sets and relationship-sets in the E-R model is so similar to that which was found in the relational model.

3.5. The Entity-Relationship Model of FIRESYS

Chen defines four steps to the construction of an E-R data base design model, and they are:

* identify the entity-sets and relationship-sets of interest.

* identify the semantic information in the relationship-sets, such as the number of entities involved in the relationships (one-to-one, one-to-many, or many-to-many), or any dependencies of entity-sets upon other entity-sets.

* define the value-sets and attributes for the entity-sets and relationship-sets.

* determine the primary keys

From this series of steps, one can construct the ERD. The entity-sets and relationship-sets of interest were determined through the construction of the relational model. The number of entities involved in the various relationships and any dependencies was determined by careful study of the users definition of the problem area. The value-sets and attributes were described by the users and were formalized to a certain degree in the development of the relational model. Finally, the primary keys were also determined for the most part in the relational model. The attempt was made in each of these steps to do an analysis from the E-R perspective even though many of the steps were very similar to those performed for the development of the relational model. Many of the concepts and goals of the two models are similar even though some names have been changed.

The net result of the E-R model is a description of the FIRESYS data structure needs. It includes the entity-sets and relationship-sets involved with some semantics about what the types and meanings of the relationships are. The E-R model of the FIRESYS project is presented in figure 3-4. The attributes and value-sets are not presented here since their contribution to the overall logical data structure is minimal. For a listing of the attributes of each major entity the

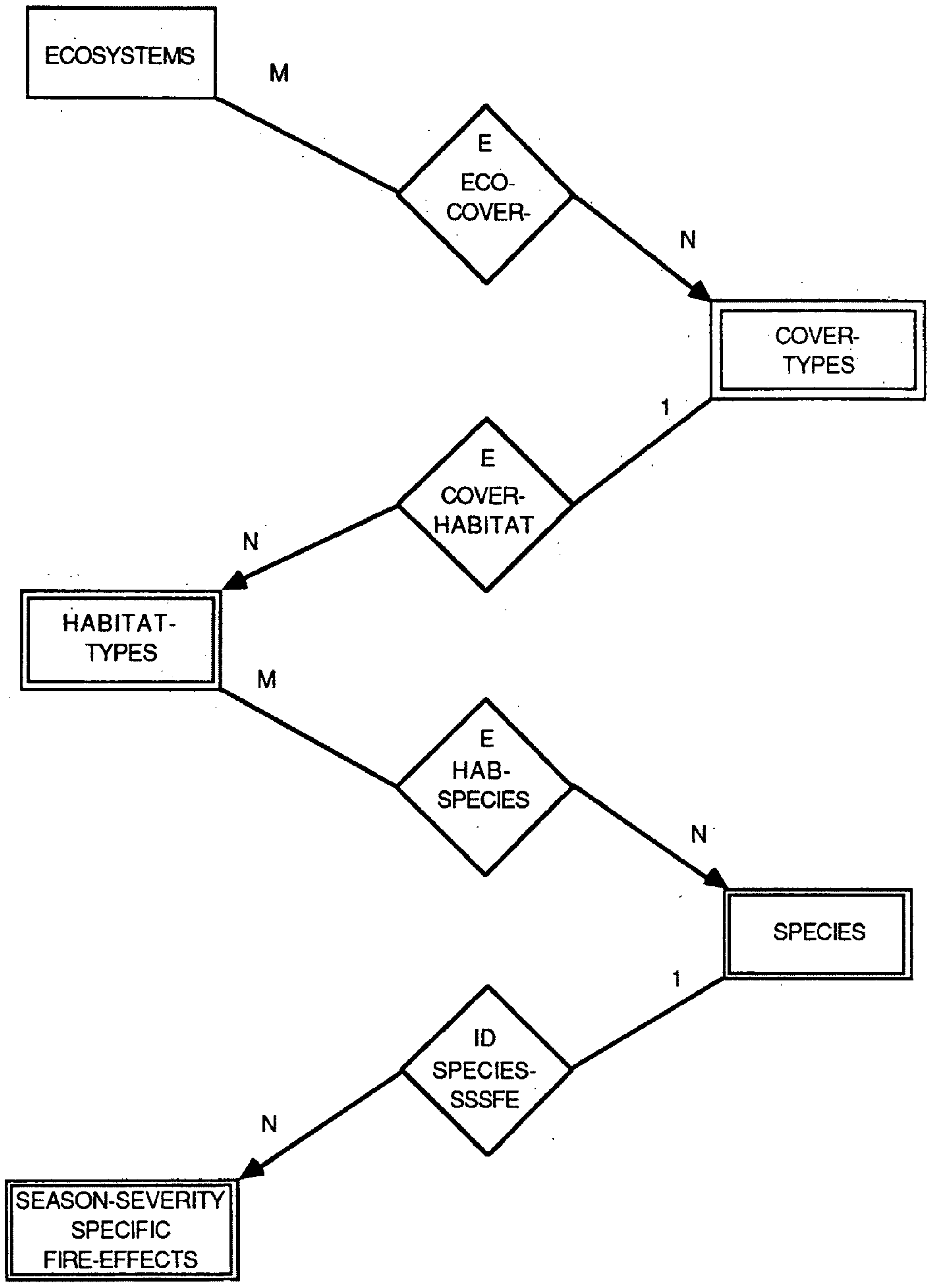


Figure 3-4: The E-R Diagram for FIRESYS

reader is directed to Appendix A.

3.6. Summary of the Entity-Relationship Model

The E-R model is a somewhat more intricate, detailed model as compared to the relational model. It is also a more powerful, descriptive model of the world that it is attempting to describe. The arrangement of entities and relationships seems to be fairly easy for most people to relate to. The average person's view of the world is usually not defined as precisely as the E-R model's is, at least this author's view is not. In that light, the E-R model may be difficult for some people to adjust to since there are so many specific definitions and concepts to deal with. Rather than just having attributes with values from a domain to describe an entity, there are value-sets, roles, attributes and domains. This author had to work hard to understand these concepts as much as was possible.

Where the relational model has a wide following as being both a model and an implementation method, the E-R model is not as widely accepted as a model, and this author knows of only one implementation based on the E-R model (Benneworth, Bishop, Turnbull, Hollman & Monette, 1981). Ng, in (Ng, 1981) goes through a process of transforming an E-R model to a physical representation so the methods have been established for the E-R model to be implemented. It may just take more time for the power of the E-R model to be appreciated and expected in the commercial data base environment.

Chapter 4

Comparison of the Relational and Entity-Relationship Models with the Existing FIRESYS Structure

The objective of this chapter is to compare the findings of chapters two and three, on the relational and entity-relationship models respectively, with a model of the actual FIRESYS system. First a model of the existing system will be presented.

4.1. A model of the Existing FIRESYS.

4.1.1. Some History of FIRESYS

There were many factors involved in the decision to model and implement FIRESYS the way it was. The first step was attempting to understand the needs of an expert system since this was the original desire of the users. There is no well defined, well accepted standard for modeling and designing an expert system since it is a relatively new area within the artificial intelligence sub-field of computer science. Certain factors were desired from The FIRESYS team point of view. These factors included the use of inferencing via inheritance in the frame representation. Also, it was felt that an object oriented environment would be beneficial to the system's functionality.

The information given to the FIRESYS team was organized into a logical understandable structure. Both a narrative listing of the objects of interest with their respective attributes and a tree structured relationship of those objects were developed.

The language to use was debated several times and it was always felt that Lisp was the strongest choice. The primary factors that favored Lisp were the growing acceptance of a common Lisp standard, the flexibility of the language including the ability to have variable length fields, and the fact that Lisp is the generally accepted language for artificial intelligence.

Our mission was to build a prototype. Due to the nature of prototyping, which include the desire to get something up and running in a short amount of time, it was known that there would not be a complete and precise specification and design for the project before the coding phase began. A model was developed which seemed workable and descriptive of the application. This model was generated in the limited amount of time available. After the coding was underway the users retracted their earlier statement of interest in an expert system. They now felt that their goal for the initial prototype was an information entry and retrieval system. It was felt by the FIRESYS team that the initial model and design concepts for the expert system were flexible enough to adapt to this new request. It was also felt that the initial model would facilitate the eventual conversion from an information system to an expert system. The decision was made by the FIRESYS team to stay with the initial model framework.

If it had been known from the very beginning that a data base would be implemented, then the relational model or the E-R model may very well have been used. The FIRESYS team contemplated the use of a commercial data base system at this stage, but there were several factors that opposed this decision. First of all, a suitable commercial program was not available that would run on the two primary target machines. The intent was to run the program on a Data General minicomputer and on IBM compatible microcomputers. Another limiting factor in the selection of a commercial data base program was the need for variable length fields. Very few data base systems provide this feature.

If a commercial data base had been found which was suitable, it most likely would have been based on the relational model. This would have lead the FIRESYS team to develop a relational model of FIRESYS in order to adapt the data to the implementation. Since a commercial data base was not found, the FIRESYS team modeled the data in what seemed to be an appropriate manner.

The model for the FIRESYS data structure had many versions during the initial phase of the project. The model being presented here is an abstraction of the system as it appeared in May of 1986. This phase in the system's life-cycle was somewhat of a milestone as the system was being presented to the commissioning personnel at the firelab. This stage was considered the final prototype resulting from their first one year grant. The system did not stay at this stage for very long as another grant was established. The system has continued to evolve to this day.

The model used for the FIRESYS development was somewhat ad hoc due to the uncertainty about how to model an expert system. The model had two components. As stated earlier, FIRESYS is a frame-based system built in Lisp and the set of frames, listing the major objects with their respective slots, was one component of the model. For a description of frames the reader is directed to (Minsky, 1985). The other component of the model was a tree structured chart showing the relationships between the various frames.

4.1.2. Frames and Slots

A frame can be thought of as a structured representation of some object or of a class of objects (Fikes & Kehler, 1985). A frame is made up of a group of slots. Slots in a frame are somewhat analogous to attributes in the relational model. Slots contain values for certain properties of the object being represented by the frame. The value in a slot may be an individual value or it may be a reference to another frame with its own slots. For example, our species frame contains a list of slots which contained the values for scientific name, abbreviated name, a list of common names and references, or pointers, to other frames. The other frames pointed to may be more detailed collections of data about the species or they may be frames with general information about a group of species. This allows frames to be linked together to form what is called a semantic net.

There is growing interest in using frames as a means of storing information. One feature for which frames are commonly used is inheritance. The idea is that a set of frames may be related in some fashion. The features common to a set of frames or the facts that relate a group of frames can be stored in one frame. Often this new frame is referred to as a superior frame. The set of subordinate frames can then inherit the properties, or values from the superior frame. Each of the subordinate frames contains a "parent" pointer to the superior frame. One can also override this inheritance from the superior frame. If a value that could be inherited from a superior frame already exists in the subordinate frame, then the value in the subordinate frame is used rather than looking in the superior frame. When the overriding value is not present in the subordinate frame, the inheritance is a means of inferring new information about an object.

The new fact being inferred is not actually stored with the object. The assumption is made that since no specific information is stored with the subordinate frame, the value in the superior frame is acceptable. Often a set of rules is used to aid this process of inferencing.

For example, let us say that a group of species all exhibit the same growth form. That is, they all grow as a low shrub which has a maximum height of eighteen inches. A superior frame can be created to represent this class of low-shrub plants and information common to all low shrubs can be stored in this one frame. Such information might include their susceptibility to wind, their use by animals for shelter or other common features. If there was no information stored with the individual species frame about its use for shelter by animals, then it could be inferred that the plant was used as it was stated in the superior low-shrub frame.

This is a very desirable trait for an expert system since one cannot store every piece of information on a subject. A good approach is to store the basic properties and details as facts, and infer any other information, by the use of inheritance and rules. The FIRESYS team suggested the use of this property of inheritance in data frames, but the users did not support this type of model. The FIRESYS team did use the inheritance property of frames very successfully at the systems level of the implementation in order to help develop an object oriented environment.

4.1.3. The Model and the Implementation

The data structure established for FIRESYS was directed to some degree by the implementation process. An incremental prototype was being built. The access paths for the data were modeled in order to facilitate this incremental approach. The first data to be entered into the system were one ecosystem and the species that were contained in that ecosystem. The user wanted to be able to access the species entities directly upon entering an ecosystem. The model of the data showed the species entity type being directly related to the ecosystem entity type. This picture gives a misleading view of the structure of the data. As shown in chapters two and three, the relationship between an ecosystem and a species is through the cover-type and habitat-type classifications.

There were some initial attempts at modeling and implementing the many-to-many relationship which existed, for example, between the species and the habitat-type entity types. Due to the users uncertainty as to what exactly they wanted and the time constraint that was in place, the decision was made to put off these relationships until an overall view showed how best to handle them.

4.1.4. The FIRESYS Model

The major data frame types with their respective, relevant slot types will be presented. The reader is directed to Appendix A for a full listing of all the frames with all their slots. This full listing of the data frames in Appendix A provides one of the two components to the FIRESYS model. The list includes the slots within each frame which contained values as well as the slots which contained pointers to other frames. The FIRESYS system also contained another set of frames which were not known to the user. These frames provided the data dictionary and the object oriented capacity of FIRESYS. These other frames enabled the system to keep track of what type of slot a given slot was when it was in use. Based on the slot's type, various actions could be performed on that slot. How those actions were carried out was part of the object-oriented environment's task. These system frames will not be dealt with in this paper.

The major objects which the users were interested in are the same as those listed in section2.4, and those were:

* Ecosystems
* Cover-types
* Habitat-types
* Species
* Season-Severity-Specific Fire Effects

Each of these objects had a frame type made up for it. The frame contained the slots which acted as the attributes for each of the objects.

An abbreviated picture of the tree structured model that was used during the early implementation of the FIRESYS system is given in figure 4-1. The user is directed to Appendix B for a more detailed diagram. By being a tree structured design, this component of the model is not capable of showing the many-to-many relationships that were desired. However, the frame-based component of the design is capable of many-to-many relationships, by the use of lists of pointers to other frames. There were intentions to utilize the capability of frames to reflect many-to-many relationships but for reasons presented earlier in this paper, this was not done.

4.2. The Relational Model vs. The Implemented FIRESYS

It has become obvious to the author that the relational model is superior to the two component model that was used by the FIRESYS team. This is not very surprising since the FIRESYS model was a rather ad hoc model.

One of the primary advantages of the relational model is its ability to represent many-to-many relationships very clearly and precisely. The HABITATTYPE-SPECIES relation in section 2.4.2 is just one example of this clarity. FIRESYS attempted to model this relationship but there was no mathematical validity to our method. Also, the relationship was not stated as explicitly as it was in the relational model.

It is easy to assume that one would follow through and utilize a relational data base for the implementation, after using the relational model. The internal concerns of how to implement many-to-many relationships is handled by the software. This is the intent of the relational model; to remove itself from the

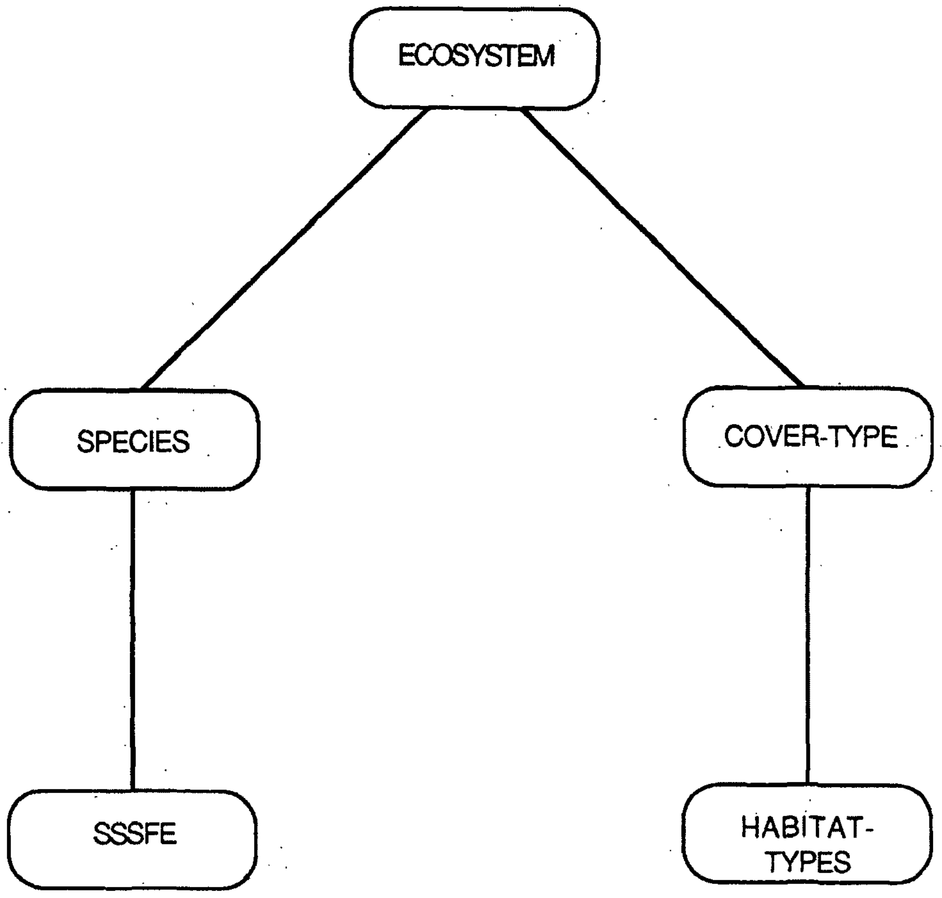


Figure 4-1: The FIRESYS Model of FIRESYS.

concerns of the implementation. The FIRESYS implementation was built from scratch. The team used a non-rigorous model which was based on unstable decisions from the users. All of these factors combined to make the implementation of many-to-many relationships much more difficult to deal with.

One interesting fact that has come out of this study is the indexing scheme used in the implementation vs. the implied indexing in the relational model. For an example, let us look at the COVER-TYPE and HABITAT-TYPE entity-types and their relationship.

First, note that this is a one-to-many relationship with one cover-type having many habitat-types and each habitat-type being in only one cover-type (this is based on the current classification system being used by the firelab personnel). Let us see what is involved when adding a new habitat-type to the model and to the implementation. It is assumed that the cover-type in which the habitat-type is found has already been stored in the data base.

When a habitat-type is added in the implementation, a pointer (the index) to that habitat-type is stored in a list of habitat-type pointers within the cover-type frame. This way, the set of habitat-types can always be located by the encompassing cover-type.

Using this same example but with the relational model, the following situation occurs. When the habitat-type is added to the data base, the name of the cover-type in which the habitat-type is found is stored as an attribute in the habitat-type tuple. The implementation version stores the index at one end of the relationship between the two items (at the cover-type end), while in the relational model the index is stored at the other end of the relationship (at the habitat-type end). Admittedly, this is comparing a model with an implementation, and this author has not determined any significance to this fact, but it is an interesting point.

Another point that became obvious during the development of the relational model is the following. It was strictly for the users conceptual benefit that all of the sub-frames under the species frame were organized as they were, rather than listing them all as slots in the species frame. There was no inherent modeling advantage to doing this. It was done simply to break down a large group of slots into smaller groups. It is most likely that the same grouping would have been done if the relational model had been used from the start. The difference would have been that with the relational model it would have been clearer exactly why this was being done.

Another factor that is very beneficial in the relational model is its ability to structure the data without any interest in the implementation of the system. The data structure model and the design and implementation of a system must be looked at as two separate although related aspects to the overall development of a major project. These two aspects of FIRESYS, the logical structure of the data and the implementation of the system, became too closely tied to one another.

In summary then, the advantages of the relational model over the FIRESYS model (if you call our model a model) are:

* The capability of the relational model to describe the logical data structure of the system, with no direct connection to the implementation needs of the system.

* The capability of the relational model to show many-to-many relationships between data items in a clear and precise manner.

* The relational model has commercial implementations that are readily available. It would have been relatively easy to generate a working prototype from the relational model in that environment, rather than constructing our own environment from scratch.

4.3. The Entity-Relationship Model vs. The Implemented FIRESYS

The E-R model of FIRESYS is a much more useful and powerful model than was the actual FIRESYS model. Much of what was said about the advantages of the relational model vs. the implementation model can also be said for the E-R model.

Use of the E-R model forces the design process to be more rigorous in terms of analyzing what the intended application for the data base will be. The users and data base designers must have no confusion or misunderstanding between them. If there is not a complete and accurate exchange of ideas about the intent of the data base, there could be incorrect decisions made during the development of the E-R model. The point raised in section 3.2.1 about whether marriage is an entity or a relationship is an example of how important it is to know exactly how the data base will be used.

What has been achieved by the use of the E-R model is a clean, precise picture of what data is being stored and what the relationships are between the data. The objects of interest to the user are classified as entity-sets. The relationships that the user feels are important are classified as relationship-sets. Important facts that pertain to the entity-sets are handled as attributes to the entity-sets. Those facts that are components of the relationship between entities are dealt with as attributes of the relationship-sets. The model is structured in a manner that is much like the users view of the data, and this is a desirable trait in a data model.

There are several advantages to using the E-R model. First is the separation of entities and relationships. This separation makes it easier to see what data items there are, and to see what the relationships are between the data items. The E-R model also shows the existence and identity dependencies. These additional semantics which are explicitly expressed in the model, help the eventual data base to more accurately represent the real world environment being modeled.

One factor that was dealt with in section 2.3.2.1 was the problem of multi-valued attributes, which is an attribute that is made up of a list of items. This involves cases such as the list of common-names for a given species. This list of common-names was treated as one item in the implementation of FIRESYS. It was recommended in the relational model that these multivalued attributes be broken down into separate tuples for each item in the list. The E-R model allows one to specify multi-valued attributes in a simple manner.

The author is concerned as to whether the drawbacks associated with treating a list of items as a single item in the relational model would also be drawbacks in the E-R model. It is very convenient to be able to specify multi-valued attributes as in the E-R model, but there must be no drawbacks to this representation. The biggest problem would be the addition of new data that is related to each of the elements in the list. A possible solution to the problem is to treat the list of items as another entity-set, rather than an attribute. This would require another relationship which would involve the species entity-set and the newly created entity-set for the listed items. Then if some new factor was associated with the listed items, it could be incorporated as an attribute in the listed item entity-set. Without knowing how the E-R model is actually implemented, it cannot be said whether this would be a problem or not. A competent data base designer must be aware that multi-valued attributes must be dealt with.

The fact that the E-R model is somewhat more complicated than some other models may be somewhat of a disadvantage to some users. However, the user does not need to understand how to set up the E-R model. The user does need to be able to read the E-R model with the help of a data modeling expert. It is the data modeling expert that needs to clearly understand how to set up and work with the E-R model. In the long run the user should feel that a better picture of his/her application has been created. This will result in a better data base implementation and this is what the user is looking for.

Overall, this author feels very good about using the E-R model for an application. It provides a rich description of the desired data base and includes explicit information that was either implicitly stated in the relational model or was not presented at all. The E-R model should become more widely accepted and used in the future due to its power and expressiveness. An increase in the acceptance of the E-R model will result in more implementations based on the E-R model.

There is still one question that has not been answered and that is whether the E-R model is capable of modeling inheritance in frames. (see section 4.1.2). There was no inheritance used in FIRESYS, so there are no examples to be examined and tested. The author cannot answer the question about whether inheritance can be handled in the E-R model, except to say that it could possibly be modeled as a relationship.

Chapter 5

Summary and Conclusion

The primary question that this author hoped to answer was whether the relational data model and/or the Entity-Relationship data model would be successful at modeling a frame-based information system. The conclusion reached is a strong yes. A frame-based information system can successfully be modeled by either the relational data model or the Entity-Relationship data model. However, this claim must be qualified. The frame-based system that was modeled does not utilize the power of inheritance. This potential to infer values based on inheritance is a major reason to use frames. The fact that inheritance was not used meant that no evaluation could be done to see if the two data models could represent this feature. The author does not feel qualified to speculate on whether either of the two models will be successful with inheritance, since inheritance was not incorporated into FIRESYS.

The system that was constructed, FIRESYS, can be modeled successfully by these two data models. This has been done in this paper, and the important points found are presented here. The more powerful data models being used today, such as the relational and E-R models, are attempting to remove the implementation concerns from the data model. The objective of these models is to clearly and precisely state what data items will be stored and what their relationships will be. How the implementation is carried out is another step in the overall development of a system. The data modeling should be done as an individual phase and the implementation should be done as another individual phase. They are each components in the life-cycle of an information system. There is a transition from one phase to the next in this life-cycle. The smoother this transition is the better the end product will be. An important factor in facilitating a smooth transition is to clearly understand the product at each phase in the life-cycle. Either model would help to provide a better final product since both the relational and the E-R models provided a better and clearer picture of the data and its structure than did the actual FIRESYS model.

In terms of which of the two models is better, there is room for debate. Let us examine the relational model first. The FIRESYS project that was implemented was relatively simple in terms of its data types and relationships. The relational model was able to unambiguously represent the data structure. There are commercially available data base programs that are based on the relational model. The use of the same model in both the modeling and implementation phases would help provide a smooth transition from the beginning to the end of the development process.

Using the relational model approach may very well be faster in terms of development and implementation time. This is due to it's simpler modeling syntax and it's availability as a commercially implemented data base. If the system never becomes an expert system, and so never required the additional modeling capabilities of the E-R model, then the relational model may be the better of the two methods.

Let us now examine the E-R model. When looking at the long range goal of FIRESYS there are many questions about how the system will be constructed. It is expected to become an expert system and this is still a somewhat experimental area in computer science. Due to the experimental nature of this field, the tools and concepts used in the expert system's development should be as powerful, expressive and adaptable as possible.

There is no inherent weakness in using the E-R model on a relatively simple domain such as the current FIRESYS. In light of the future intentions for FIRESYS, the E-R model would have the additional features that may be needed to represent more complicated structures of the data and it's relationships. For this reason, this author feels that it may be helpful for the E-R model to be incorporated into the FIRESYS project.

The disadvantages of including the E-R model into FIRESYS would include the work of maintaining an additional model, and the lack of correlation between the E-R model and the current implementation. This lack of correlation will probably exist for several years. The implementations based on the relational model are only now, sixteen years after the introduction of the relation model, being accepted as valid, effecient programs. Actully, Codd the father of the relational model of data bases still feels that there is not one current implementation that fully reflects the relational data base model (Codd, 1985). This implies that it will be a number of years before there are fully acceptable implementations based on the E-R model since the E-R model has only been out for about 10 years.

The advantages of adding the E-R model to the FIRESYS project include its ability to present a more complete picture of the data. The separation of entities and relationships as explicit components of the model as well as the expression of dependencies in the model, provide a better understanding of the data. Also, assuming that the FIRESYS team continues to develop their own code, they have the freedom to attempt to incorporate some of the expressive power of the E-R model directly into the implementation. There is at least one data base management system available, called GERM, that is based on the E-R model (Benneworth, Bishop, Turnbull, Hollman & Monette, 1981). This is another option for the FIRESYS project to consider.

There is an area of FIRESYS that this paper has not addressed and will not address in any detail. This is the concept of an object-oriented programming environment. During the research for this paper there were no references found that indicated any use of the E-R model or the relational model within object-oriented environments. The problem of integrating a relational model or E-R model with the object-oriented programming paradigm is an open problem thatthis paper does not attempt to deal with.

One very important concept that resulted from this study is that of the separation of the data modeling from the design and implementation of the system. It is vital to the data base or information management system that the data's structure be very clearly understood. If there are mistakes in the representation of the data, it will not matter how good the implementation is. The system will not reflect what the user desires. If the structure of the data is correct and includes valid semantics about the data and its relationships, then the implementation has a much better chance of satisfying the end user.

In conclusion, the representation of the data in FIRESYS was moderately accurate when taking into account the circumstances under which the FIRESYS project was developed. If it had been clearer from the beginning what was desired from FIRESYS and if the decision had been made to use a data modeling technique such as the relational or Entity-Relationship model, then a much better representation of the data would have been possible. The relational and Entity-Relationship data models do work well with frame representations of data. This statement assumes that no inferencing by inheritance is involved in the system. This factor of inheritance was not examined during this study.

Data modeling must be done prior to implementing an information management system. If it is not done there is a very good chance that the system will not accurately reflect the user's logical view of the data. If a thorough, clear and accurate data model is developed, there is a much better chance that the final system will meet the user's expectations. Meeting or exceeding the user's expectations should be the goal of any software development project.

Appendix A

List of Entities and Attributes in FIRESYS

species/entity

SPECIES
ABBREVIATION
SCIENTIFIC-ALIAS
COMMON-NAMES
LIFE-FORM
VARIETIES-AND-FORMS
FIRE-EFFECTS
HABITAT-TYPES

The following indented sections are directly related to the species entity, but we are showing the sub-groupings that have been established.

distribution-and-occurrence/entity

SPECIES
GENERAL-DISTRIBUTION
BLM-PHYSIOGRAPHIC-REGIONS
KUCHLER-PLANT-ASSOCIATIONS
SAF-COVER-TYPES
HABITAT-TYPE-INFORMATION
REFERENCES

value-and-use/entity

```
SPECIES
DESCRIPTION
PALATABILITY
FOOD-VALUE
COVER-VALUE
IMPORTANCE-TO-LIVESTOCK-AND-WILDLIFE
OTHER-USES-AND-VALUES
ENVIRONMENTAL-CONSIDERATIONS
REFERENCES
```

botanical-and-ecological-characteristics/entity

```
SPECIES
GENERAL-DESCRIPTION
GROWTH-FORM
RAUNKIAER-LIFE-FORM
GRIME-PLANT-STRATEGY-CLASSIFICATION
GRIME-REGENERATIVE-STRATEGY-CLASSIFICATION
REGENERATION-PROCESSES
SITE-CHARACTERISTICS
SUCCESSIONAL-STATUS
SEASONAL-DEVELOPMENT
REFERENCES
```

fire-adaptive-traits-and-survival-strategies/entity

```
SPECIES
DESCRIPTION
LYON-STICKNEY-FIRE-SURVIVAL-STRATEGY
NOBLE-AND-SLATYER-VITAL-ATTRIBUTES
  SPECIES-TYPE
  TIME-UNTIL-MATURITY
  TIME-UNTIL-SENESCENCE
  TIME-UNTIL-EXTINCTION
ROWE-MODE-OF-PERSISTANCE
REFERENCES
```

fire-effects/entity

SPECIES
FIRE-EFFECT-ON-PLANT
DISCUSSION-AND-QUALIFICATION-OF-FIRE-EFFECT
PLANT-RESPONSE-TO-FIRE
DISCUSSION-AND-QUALIFICATION-OF-PLANT-RESPONSE
SEVERITY-SEASON-SPECIFIC-FIRE-EFFECTS
REFERENCES

severity-season-specific-fire-effects/entity

SPECIES
SEVERITY
SEASON
EFFECT
CERTAINTY-FACTOR
DESCRIPTION
QUALIFICATION
REFERENCES

This concludes the entities that are grouped
with the species entities.

;;
ecosystem/entity

CLASSIFICATION-KEY
FOREST-AND-RANGE-ENVIRONMENTAL-STUDY-FRES-NUMBER
KUECHLER-VEGETATION-TYPES
DISTRIBUTION
SITE-CHARACTERISTICS
SOILS
CLIMATE
COVER-TYPES
REFERENCES

cover-type/entity

COVER-TYPE
ECOSYSTEMS
CLASSIFICATION-KEY
ABBREVIATION
DISTRIBUTION
SITE-CHARACTERISTICS
VEGETATIVE-COMPOSITION
TREES
SHRUBS
GRASSES
FORBS
OTHER
SUCCESSIONAL-TRENDS
HABITAT-TYPES
REFERENCES

habitat-type/entity

HABITAT-TYPE
COVER-TYPE
CLASSIFICATION-KEY
ABBREVIATION
DISTRIBUTION
SITE-CHARACTERISTICS
VEGETATIVE-COMPOSITION
TREES
SHRUBS
GRASSES
FORBS
OTHER
SPECIES
INDICATORS-OF-GOOD-CONDITION
INDICATORS-OF-POOR-CONDITION
SUCCESSIONAL-TRENDS
HABITAT-MANAGEMENT-CONSIDERATIONS
HABITAT-FIRE-ECOLOGY-AND-EFFECTS

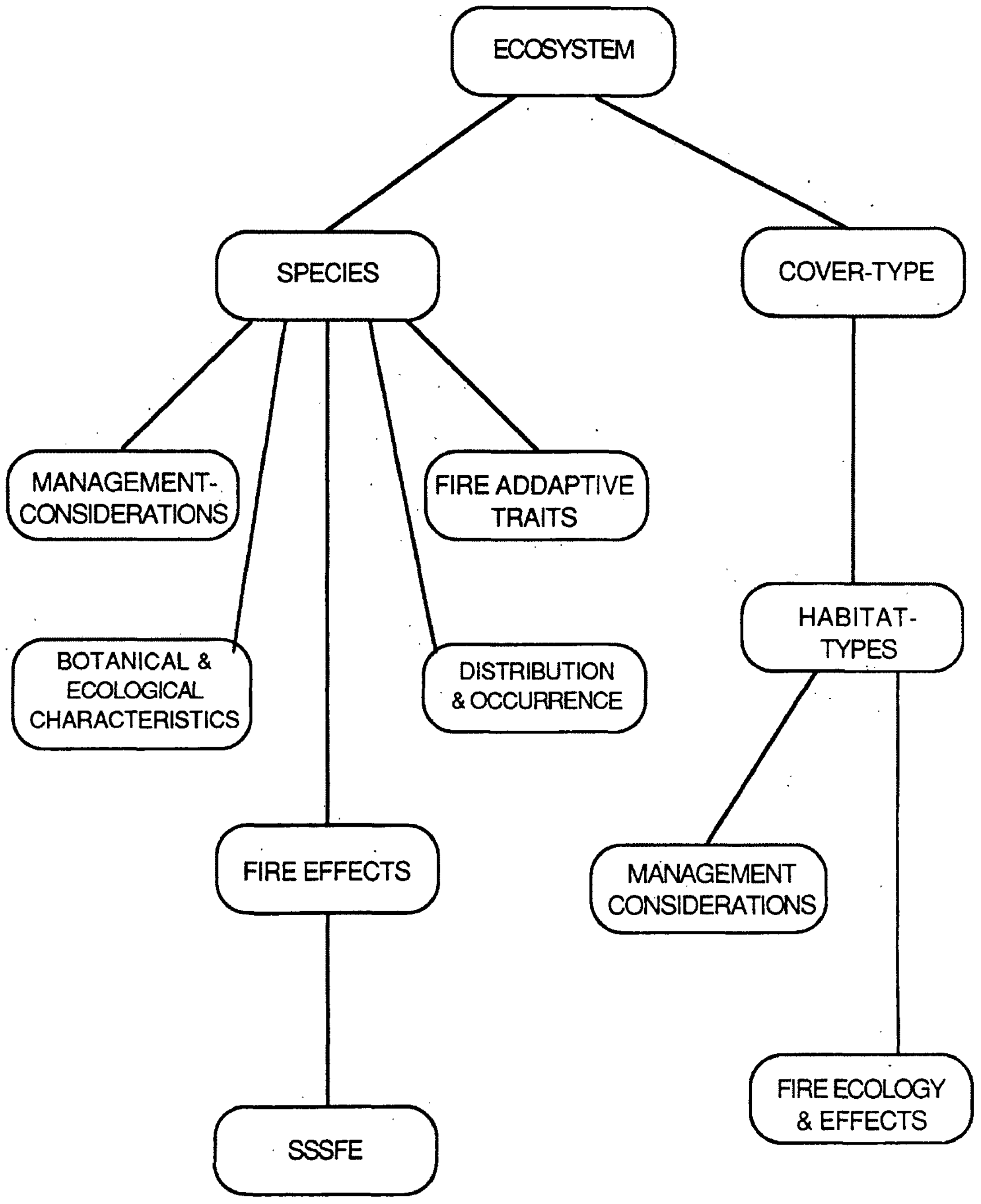
habitat-management-considerations/entity

LIVESTOCK-RANGE
WILDLIFE-HABITAT
OTHER-HABITAT-CONSIDERATIONS
REFERENCES

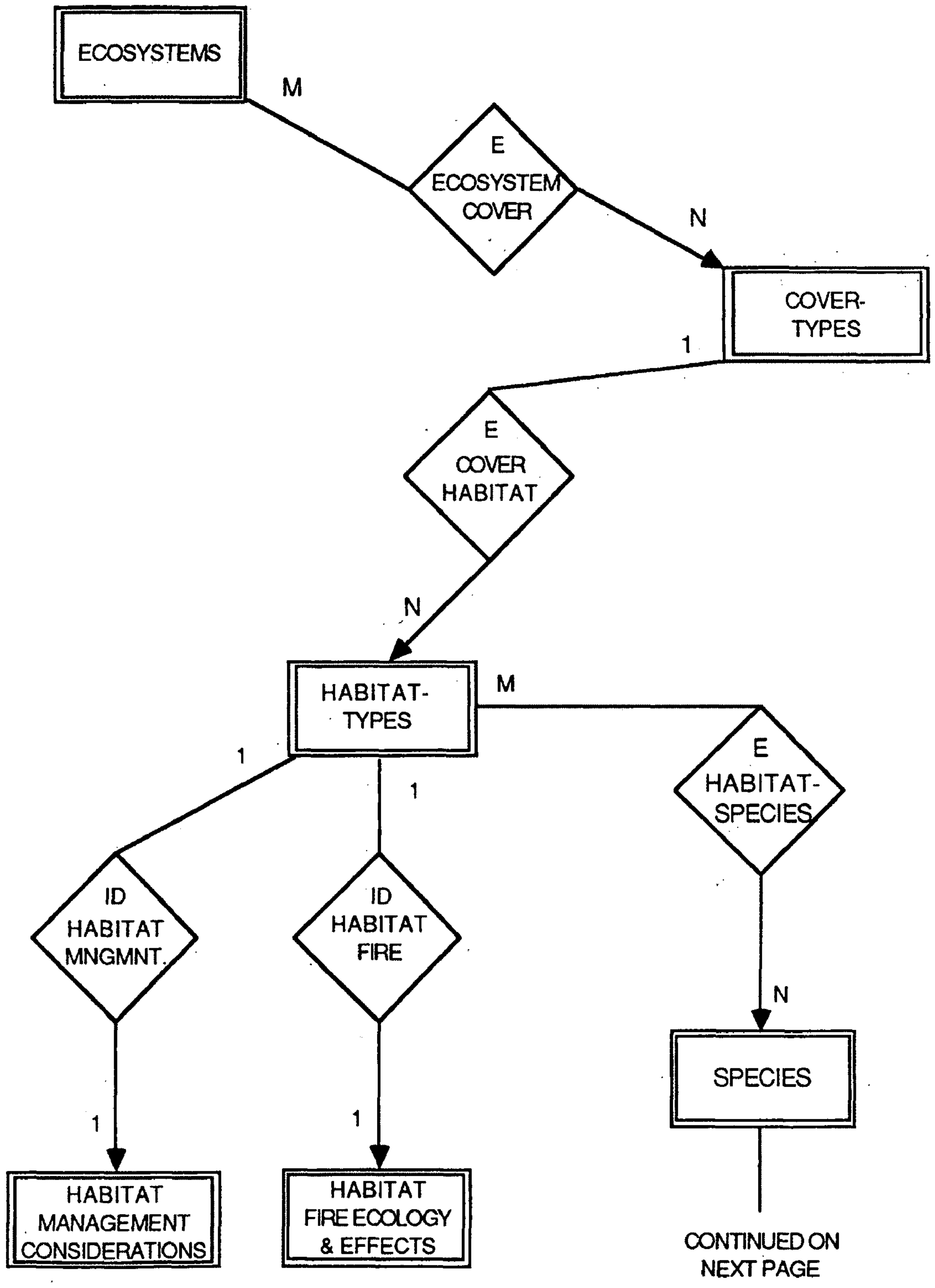
habitat-fire-ecology-and-effects/entity

FIRE-OCCURRENCE
IMMEDIATE-FIRE-EFFECTS-ON-COMMUNITY
IMMEDIATE-COMMUNITY-RESPONSE-TO-FIRE
LONG-TERM-COMMUNITY-RESPONSE-TO-FIRE
FIRE-EFFECTS-ON-GRAZING-POTENTIAL
FIRE-EFFECTS-ON-WILDLIFE-HABITAT-AND-POPULATIONS
FIRE-USE-POTENTIAL
REFERENCES

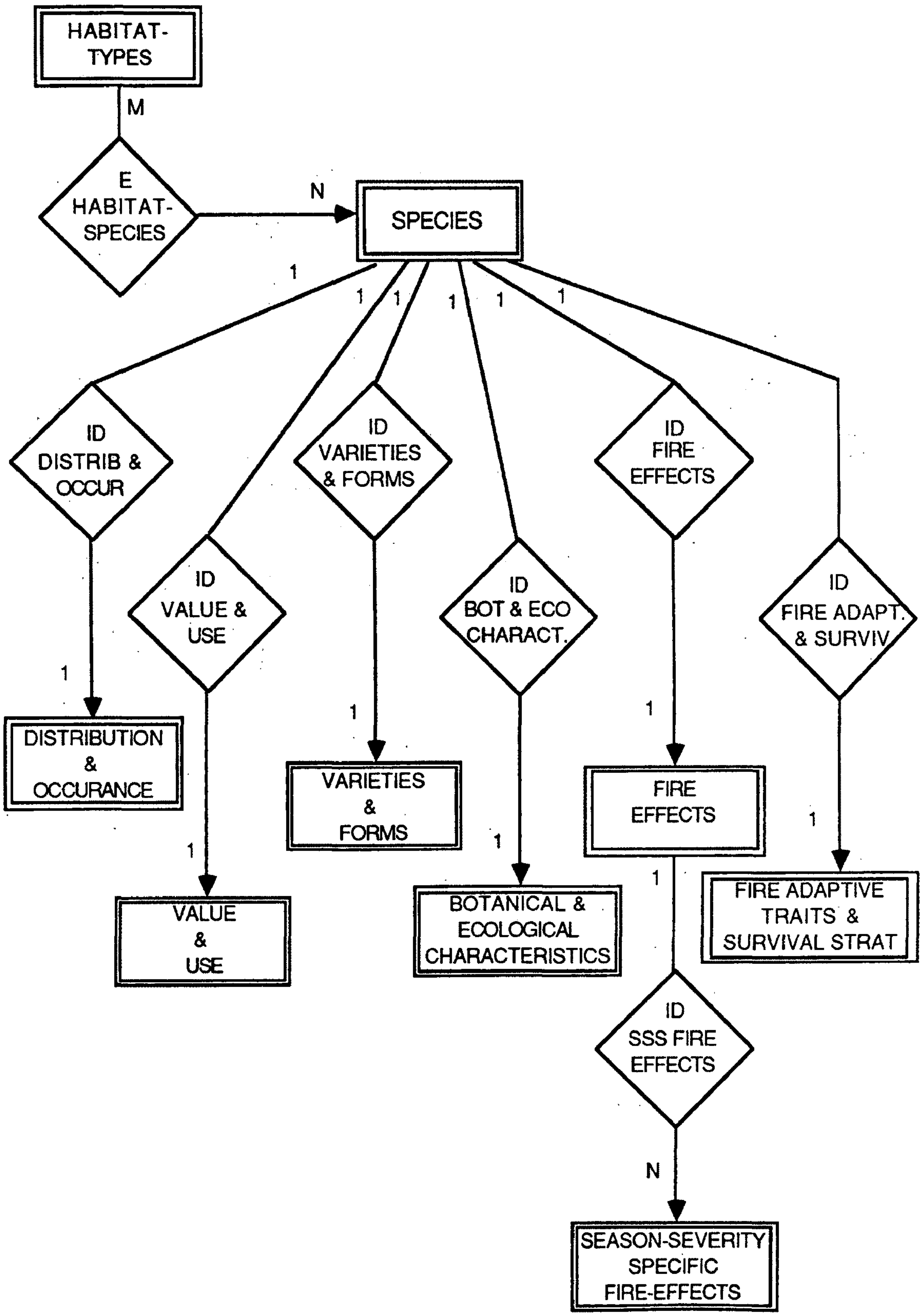
APPENDIX B. The Frame-Based Hierarchy of FIRESYS


ECOSYSTEM
SPECIES
COVER-TYPE
MANAGEMENT-CONSIDERATIONS
FIRE ADDAPTIVE TRAITS
BOTANICAL & ECOLOGICAL CHARACTERISTICS
DISTRIBUTION & OCCURRENCE
HABITAT-TYPES
FIRE EFFECTS
MANAGEMENT CONSIDERATIONS
FIRE ECOLOGY & EFFECTS
SSSFE

APPENDIX C. The ERD of FIRESYS



APPENDIX C The ERD of FIRESYS cont.



Bibliography

Benneworth, R.L. & Biship, C.D. & Turnbull, C.J.M. & Holman, W.D. & Monette, F.M. *The implementation of GERM, An Entity-Relationship Data Base Management System*, pages 478-484. IEEE Proceedings of International Conference on Very Large Data Bases, 1981.

Bic,. Lubomir & Gilbert, Jonathan P. Learning from AI: New Trends in Database Technology. *IEEE Computer*, Mar 1986, *19(3)*, 44-54.

Carlson, C. R. & Arora, A. K. Toward the Next Generation of Data Modeling Tools. *IEEE Transactions on Software Engineering*, Sept 1985, *SE-11(9)*, 966-970.

Chen, Peter P-S. The Entity-Relationship Model--Toward a Unified View of Data. *ACM Transactions on Database Systems*, March 1976, *1(1)*, 9-36.

CODASYL. *CODASYL Data Description Language Journal of Development.* National Bureau of Standards Handbook 113; U.S. Gov't Printing Office, 1973. COmmittee on DAta SYstems Languages.

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

Codd, E. F. Does your DBMS run by the rules? *Computerworld*, Oct. 21, 1985, }, 49-60.

Fikes, Richard & Kehler, Tom. The Role of Frame-Based Representation in Reasoning. *Communications of the ACM*, Sept 1985, *28(9)*, 904-920.

Hartzband, David j. & Maryanski, Fred J. Enhancing Knowledge Representation in Engineering Databases. *IEEE COMPUTER*, Sept 1985, *18(9)*, 39-46.

Maier, David. *The Theory of Relational Databases.* Computer Science Press, 1983.

Martin, James. *Computer Data-Base Organization.* Prentice-Hall, 1975.

Martin, James. *Principles of Data-Base Management.* Prentice-Hall, 1976.

Minsky, Marvin. A Framework for Representing Knowledge. *Readings in Knowledge Representation*, 1985, }, 245-262.

Ng, Peter A. Further Analysis of the Entity-Relationship Approach to Database

Design. *IEEE Transactions on Software Engineering*, January 1981, *SE-7(1)*, 85-99.

Tsichritzis, D. & Lochovsky, F. *Data Models.* Prentice-Hall, 1982.