

INCORPORATING EXPERT SYSTEM TECHNOLOGY
INTO A PROFESSIONAL GENEALOGICAL
INFORMATION SYSTEM

by

BARBARA STOUT PARKER

B.S., Virginia Tech University, 1978

A MASTERS THESIS

submitted in partial fulfillment of the

requirements for the degree


MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Approved by:


Major Professor

11
21668
174
C.H.S.C.
1988
P37
C.2

TABLE OF CONTENTS

CHAPTER 1. Overview 1

1.1 Introduction 1

1.2 Literature Review - Overview 3

1.3 Expert Systems 4

 1.3.1 Background 4

 1.3.2 Expert System Overview 5

 1.3.3 Knowledge Representation 7

 1.3.4 Knowledge Acquisition 8

 1.3.5 Expert System Tools 10

 1.3.6 Future Trends 11

1.4 Genealogy and Expert Systems 11

1.5 Database Technology and Expert Systems 14

1.6 Summary 18

CHAPTER 2. GENEALOGICAL PROCESS 19

2.1 Introduction 19

2.2 The Genealogical Hobbyist 19

2.3 The Professional Genealogist 27

 2.3.1 The Professional Approach 27

 2.3.2 Vital Records 28

 2.3.3 Census Records 30

 2.3.4 Wills 31

 2.3.5 Estate Records 34

 2.3.6 Land Records 34

 2.3.7 Tax and Court 35

 2.3.8 Professional Genealogist Summary 35

2.4 Current Genealogical Systems 36

2.5 Conclusion 38

CHAPTER 3. REQUIREMENTS 39

3.1 Introduction 39

3.2 General Requirements 41

3.3 Specific Requirements 42

3.3.1	The User Front-end43
3.3.2	The Expert System43
CHAPTER 4. DESIGN47
4.1	Introduction47
4.2	Overview of the Genealogical Information System . .48	
4.2.1	Introduction48
4.2.2	System Overview50
4.3	Scope of the Design59
4.4	Experts: Knowledgeable Modules60
4.4.1	Background60
4.4.2	What is an Expert?62
4.4.3	Decisions on a Datum63
4.4.4	Types of Experts64
4.5	The Expert/Database Interface69
4.5.1	Data Experts69
4.5.1.1	Adding Data Items69
4.5.1.2	Updating Data Items71
4.5.1.3	Deleting Data Items73
4.5.2	Establishing Identities and Determining Relationships74
4.5.3	A Sample Scenario74
4.6	Example of an Expert77
4.7	Summary80
CHAPTER 5. DESIGN PROBLEMS ENCOUNTERED82
5.1	Introduction82
5.2	Original Expert System Design82
5.3	Problems Encountered During Design Phase.86
5.3.1	Large and Complex Knowledge Base86
5.3.2	Personal Computer Requirement89
5.4	Summary90
CHAPTER 6. IMPLEMENTATION92
6.1	Introduction92
6.2	Data Base Design92

6.3	Prototype Overview98
6.4	Prototype Functions	100
6.4.1	Help Information Function	100
6.4.2	Add Source Record Function	100
6.4.3	Delete Source Record Function	103
6.4.4	Update Source Record Function	105
6.5	Summary	109
CHAPTER 7. CONCLUSIONS AND FUTURE ENHANCEMENTS . . .		111
7.1	Introduction	111
7.2	Conclusions	111
7.2.1	Expert System Design	111
7.2.2	Prototype Implementation	113
7.3	Extensions	114
7.3.1	Expert System Design	114
7.3.2	Prototype Implementation	117
7.4	Summary	117
REFERENCES		119
APPENDIX A		121

LIST OF FIGURES

Figure 2.1	Example of an Ancestral Chart	23
Figure 2.2	Example of a Family Group Sheet	24
Figure 2.3	Example of a Genealogy Lineage Chart	25
Figure 3.1	The Professional Genealogical Information System	40
Figure 4.1	Relationship of the Genealogical Information System Components	49
Figure 4.2	Overview of User Front-end	52
Figure 4.3	Relationship Between Data Experts and the Database Sub-system	58
Figure 4.4	Functional Hierarchy of the Expert Sub-system	67
Figure 4.5	Internal Expert Hierarchy	68
Figure 5.1	Original System Structure	84

LIST OF TABLES

Table 4.1	Sample of Source Record Fields	66
Table 6.1	Data Base Structures	96
Table 6.2	Data Base Index Structures	97

This thesis is dedicated to

my mother, Joan R. Minter, whose
love for genealogy inspired this
research . . .

my husband, Harold A. Parker, whose
endless patience enabled me to devote
time to this endeavor . . .and to

my professor, Beth Unger, whose
warmth and sense of humor helped me
to maintain my sanity.

CHAPTER 1 - OVERVIEW

1.1 Introduction

This thesis addresses the incorporation of expert system technology into a database system. Professional genealogical research was chosen as the problem domain because it represents a narrow, specialized area in which expert knowledge can be obtained.

Genealogy is the nation's third most popular hobby. It is in vogue to discover one's family roots. Genealogical hobbyists collect their ancestors' vital data, producing pedigree (ancestral), or descendant charts, as well as individual and family group sheets. These novice genealogists generally proceed in a hit-or-miss fashion because of limited genealogical knowledge and expertise.

In comparison, professional genealogy involves comprehensive and systematic research. This results in a historical chronicle of a family in text format, breathing life into the family lineage. The professional genealogist must be knowledgeable about his/her state's laws, records, history and culture. The genealogist is hired to research family lineages. It is imperative that the accuracy level of the findings be high. The research must be supplemented with sufficient proof of the lineage. Competent genealogical research involves long-range

strategies for combining and using all available resources. Most professional genealogists are certified by a state certification board.

The professional genealogist researches a vast amount of data such as wills, censuses, estate, tax, and vital records for clues that will unravel the family lineage. Decisions on each piece of data must be made, requiring a great deal of time and advanced genealogical knowledge. Furthermore, this data, often incomplete, must be continually compiled and reorganized to establish descendant linkage as it is collected. The successful result of this investigation is a text describing the descent and history of a family.

The genealogical process would be greatly enhanced by a database system that incorporates rules and inference provided by genealogy experts, facilitating not only the storage, linkage, and manipulation of the data, but also assisting in the genealogical analysis and decision process.

This thesis documents a project whose objective was to design and implement a system that captures the knowledge of a professional genealogical expert, stores this knowledge in a high level representation, and utilizes this knowledge in assisting the genealogist with decisions

and strategies. This system is part of a larger research project being conducted at Kansas State University. The objective of the larger project is to develop a complete database system that assists professional genealogists in their research and publishing endeavors.

A literature review follows this section, addressing expert system technology. This technology is used as a basis for a brief discussion of genealogy and database systems.

The remainder of the report describes the development of the system. Chapter 2 discusses the genealogical process. Chapter 3 identifies the requirements for the system and is followed by a description of its design in Chapter 4. The implementation and testing of the prototype software are documented in Chapter 5. The thesis concludes with Chapter 6 which presents conclusions and future enhancements to the project.

1.2 Literature Review - Overview

A literature review is presented which briefly describes existing expert system theory. Expert system research is in its infancy. The literature presented highlights the different aspects of designing an expert system. Currently, expert system technology has not been applied to the problem domain of genealogy. A rational for

choosing genealogy for this project is given in section 1.4, followed by a discussion of current research on the incorporation of expert system technology in a database system in section 1.5.

1.3 Expert Systems

1.3.1 Background

Artificial Intelligence (AI) is the field of computer science that includes a number of advanced technologies sharing a common theme of assisting humans with processes and decisions. The goal of AI is to incorporate into these technologies the ability to simulate the human thought process, intelligence. These technologies include systems which interpret visual images, recognize language, solve problems, and learn from experience [BA85,WA86].

AI research has endeavored since its infancy to develop systems that could "think". Expert systems evolved from this research. Early expert systems embraced general methods of problem solving that could be used for a broad spectrum of problems. Specialized procedural knowledge was not stored in order to retain generality. As a result, these systems were inefficient and used a high amount of computer resources. This early effort led to the development of specialized systems that used extensive, specific knowledge about narrow problem areas.

These systems use fewer computer resources but require a higher quantity of man hours and intellectual effort [WK86].

1.3.2 Expert System Overview

It is generally agreed that expert systems are computer programs that incorporate the factual knowledge of a human expert to solve problems [BA85,WA86,HR83]. Walker uses the terms knowledge system, knowledge-based system, and expert system interchangeably [WK86]. Hartzband and Maryanski define knowledge-based systems to be composed of an expert system and a database [HA85]. Waterman states that an expert system is a subset of a knowledge-based system [WA86]. In this paper, the later concept will be used.

The process of building an expert system is called knowledge engineering. This process involves a knowledge engineer, who is the expert system builder, and a human expert(s). The knowledge engineer through continual interviews with the expert, extracts the knowledge that becomes the knowledge base of the expert system. This knowledge includes rules of thumb, strategies, and the procedures of the domain area. This acquisition of knowledge is a slow repetitive process [BR83].

The ideal components found in an expert system are [WK86]:

1. facts in the form of rules (domain knowledge),
2. an inference engine (procedural knowledge),
3. an explanation generator,
4. a knowledge acquisition engine, and
5. a natural language processor.

In an expert system there are two separate bodies of knowledge, the domain knowledge and the procedural knowledge. The domain knowledge is comprised of the facts and rules obtained from the expert. The procedural knowledge comprises what is known as the inference engine. This knowledge provides the capability to manipulate the domain knowledge [WK86,WA86].

The explanation generator provides explanations on how solutions are reached and describes the processes that were used to reach them. The knowledge acquisition engine facilitates the transfer and transformation of the knowledge from the expert. A natural language processor provides a user-friendly front-end to the system [WK86,WA86].

An expert system, if chosen for appropriate problem areas, can provide numerous potential benefits. It can assist in decision making processes, save human time and effort,

provide a training tool, be used in hostile environments, and be easily documented [WA86,BA85,HR83].

Expert systems are currently being used in many of the science fields. The medical field has several expert systems in operation. Chemistry, computer systems, electronics, engineering, geology, and the military are problem domains in which expert system research activity is occurring [WA86].

1.3.3 Knowledge Representation

Knowledge representation is a data model in which information at a higher level of abstraction than the domain data is expressed. Knowledge in an expert system must be represented in such a way that the procedural knowledge is separated from the domain knowledge. The representation method should also support efficient problem solving. Generally, an expert system is structured on one or a combination of the following standard representation methods [ST83,WA86].

Rule-based systems encode knowledge as IF condition THEN action statements. Rules provide a natural way to describe processes. The rules used to reach a conclusion form an inference chain [WA86].

Frame-based systems collect all related information into a "form" or "frame", with a slot for each piece. The frames may be arranged hierarchically [WK86]. Frames are effective in representing complex information [WA86].

Semantic-net systems use a network of nodes connected by relations. These nodes represent objects, concepts, or events. Semantic nodes are also arranged hierarchically, with lower level nodes inheriting the properties of the higher levels. This representation is based on a network structure [WA86].

Mathematical logic is the final method of representing knowledge. Logic programming enables knowledge to be treated as logical theorems. Logic systems use theorem proving techniques to arrive at results. A new development has been the use of logic programming and PROLOG to produce expert systems integrated into database systems [WK86].

1.3.4 Knowledge Acquisition

The bottleneck in building expert systems is knowledge acquisition. An expert system requires highly detailed, refined domain-specific knowledge. Acquiring knowledge from an expert and transforming it to a useful form is a long, iterative process. The interaction between the knowledge engineer and the expert is critical. When

experts solve problems in their area of expertise, they often solve problems with intuitive knowledge. The knowledge engineer has to acquire this knowledge and transform it into general principles and deductive steps [WA86].

It has been suggested that knowledge engineers go through five major stages before producing an expert system. In the identification stage, the problem characteristics are identified. This is followed by the conceptualization stage in which concepts are found that can be used to represent the knowledge. The formalization stage entails designing the structure to organize the knowledge. Rules are formulated to embody the knowledge in the implementation stage. The final stage, testing, involves validating the rules that were chosen to organize the knowledge. These stages form an iterative process that refines the knowledge base [BU83].

Automated knowledge acquisition programs are being developed to ease the transfer of knowledge from human experts to an expert system. These programs fall into three forms of machine learning: learning by being told, learning by induction and examples, and learning by observation. To date, automated acquisition programs have been used for some specialized cases. This transfer and transformation of knowledge, for most cases, still

requires communication between a human expert and knowledge engineer [WK86].

Several difficulties exist in acquiring knowledge for an expert system. One difficulty is a mismatch in the representation between the way the human experts express knowledge and the form it is represented in the program. Another difficulty is our inability to express the knowledge we possess. Others include the limits on expert system technology and the complexity of testing and refining the system [BU83].

1.3.5 Expert System Tools

Expert system tools are programming languages that are used to build expert systems. They can be divided into four general categories. The programming languages are normally symbol-manipulation languages such as PROLOG and LISP. Knowledge engineering languages consist of an expert system building language integrated into a support environment. They are usually categorized as skeletal or general purpose systems. System building aids consist of automated knowledge acquisition and design systems. Support facilities consist of tools for helping with programming and providing explanations [WA86].

1.3.6 Future Trends

Expert system technology is developing rapidly. It is forecasted that in a few years, many companies will be involved with AI or expert systems. A by-product of this development will be the market for codified knowledge. Another development will be integrated expert systems. The system will be embedded in microprocessor chips, with the language and the computer in one unit. These integrated systems will form intelligent systems [DU81].

1.4 Genealogy and Expert Systems

Genealogical research is an art and a science. It is more than merely searching in records and adding names. It demands a well-planned strategy in which every possibility of finding facts and evidence is exhausted. The genealogist must have a clear definition of the research problem. What the researcher wants to know will determine what records are used, the order they are searched, and the analysis of the information they contain [H071,AL80]. Within the umbrella of the long-range plan, each search for a piece of supporting evidence should have an associated strategy on how to find and use it. The overall plan must be flexible in order to include these additional strategies. Inferences can be made from the collected research, providing clues and leads for further

searches. Allowances for adjustments must be made as the research progresses and new research directions present themselves [AL80].

It is imperative for the genealogist to have an understanding that ancestors must be kept in the context of their time period, locality, and historical setting. The records documenting these ancestors were kept according to the laws in existence at the time. The county, state, and federal laws during the time in question provide necessary clues [AL80]. For example, suppose we are trying to verify the existence of a North Carolinian named John Smith who was supposedly sixteen in 1901. We discover a 1901 land deed showing a John Smith sold a piece of property. Because we know that from 1850 to 1976, you had to be at least 21 years of age to sell land, we deduce that this is not the John Smith we are searching for. As the time and locality of a problem changes, so must the approach to that problem.

It is clear that decisions must be made and new strategies set up as diverse evidence is gathered. True evaluations and correct interpretations of the evidence are essential. Without a strong genealogical background and a carefully laid out strategy, the possibility of "twisted pedigrees" exists.

Thus, an expert knowledge of genealogy and the laws and history of the state and county are critical to an accurate ancestral chronicle. Currently, the only way to acquire this knowledge and experience is through many years of dedicated research. Jo White Linn, past President of the North Carolina Genealogical Society, states that there is a lack of available certified genealogy instructors and a lack of adequate teaching materials [AL80]. Derek Holland, a professional genealogist of the Research Department of the Genealogical Society, Salt Lake City, has the same concern. He feels there is a need to determine a method to distribute genealogical research knowledge. Because of the increase for genealogical research, expert guidance is needed on a wider scale than ever before. Instruction on where to look for data and how to weigh this evidence with careful, judgmental skill (in an orderly fashion) is lacking [HO71].

Genealogy presents a problem domain upon which expert system technology could be applied. It meets most of the "possible, justified, and appropriate" criteria established by Donald A. Waterman [WA86].

Helen F. M. Leary is a professional certified genealogist and instructor in Raliegh, North Carolina. She is the editor and a contributing author of the book, North

Carolina Research Genealogy and Local History [AL80] and is the expert who will contribute the wealth of genealogical information upon which this proposed expert system will be based [LE86].

It must be noted that there are over fifty commercial genealogy computer systems available for personal computers. These systems differ in function, capacity, and capability. Some of the current packages have been written by novices not well versed in correct genealogical record keeping procedures [KS85,SK84]. None of the currently available systems incorporate genealogical knowledge. Nor, do these systems facilitate the storage and manipulation of data, or assist in the decision and learning processes. For a more detailed analysis of current genealogical computer systems, refer to chapter 2.

1.5 Database Technology and Expert Systems

The voluminous genealogical data that must be efficiently stored, retrieved, manipulated, and linked necessitates the use of a database management system. The shortcomings of existing genealogical database systems have emphasized the need for the development of a more expressive database system. Non-computer type genealogists need a "friendlier" interface with which to communicate. They require a system which would assist them in making

intelligent queries, developing research strategies, and making judgments on pieces of data. These are among the needs that have been the impetus for incorporating expert system technology into a genealogical database system.

Data models were developed from the need to organize data. Early models included the relational, hierarchical, or the network model, all of which are record oriented. Record models arrange and store the data records efficiently, yet tend to be "non-friendly" toward users. Of the three, the relational model is the most user oriented, and allows multiple user views. It is organized into tables which represent relations, with rows that are called tuples and columns that represent attributes. The values that may appear in the columns are from its domain. Relational models are based on set theory. It is this mathematical formalism which facilitates high level, friendly queries. A relational structure allows users to have multiple views of a base relation [BI86, KR83]. Each of the above record models added to the development of database theory, yet did not have the ability to add semantics to the stored data. This need led to the development of more user oriented semantic data models such as the entity-relationship model, semantic data model, and extended relational models [BI86].

Database researchers are now investigating the incorporation of AI techniques into database models.

Semantics can be built into a database system in one of two methods:

1. Build an external model, separate from the database system which contains rules that interpret the meaning and interrelationships of data.
2. Build the semantics into the database system, resulting in a more powerful semantic model. [HA85,BI86]

Both of these approaches are in their infancy.

In an expert/database system, knowledge that is at a higher level of abstraction than the domain data is managed. This abstraction can be achieved by a modification to the set of operators, to the representation of the data, to the set of integrity rules, or a combination of the above [HA85]. Wiederhold has proposed nine categories (in decreasing abstraction) where knowledge may be applied [WI84]:

1. enterprise-directing knowledge,
2. focus management knowledge,
3. application specific knowledge,
4. general procedural knowledge,

5. structural knowledge,
6. derived knowledge,
7. data-domain knowledge,
8. knowledge about the system (query/response),
9. resiliency and recovery support knowledge.

This abstracted knowledge can be expressed by production rules, frames, semantic nets and/or mathematical logic.

The areas where AI technology, expert systems in particular, can be incorporated are in ad hoc database searching, natural language facilities, the inference engine, and rule based query support and inclusion of PROLOG [BA85].

The genealogical database with which this thesis proposes to incorporate with an expert system will be a relational model [BE86]. Genealogists must contend with unknown and incomplete data (null values). Null values in a relational data base model introduce anomalies. Thirteen types of nulls have been proposed, yet only two have been widely accepted. These include 'no currently known value,' and 'no value is valid for this attribute.' Maria Wilson's master's report presents an approach to the handling of this problem [WI85]. Artificial intelligence has been used to solve integrity problems caused by the inclusion of nulls.

1.6 Summary

The objective of this thesis is to design, and implement an expert system which facilitates a dialogue with a user and interacts with a structured database. The expert knowledge is structured into packets of rules, with logical break points established, allowing the user to modify the decision process. The system stores this knowledge in a high level representation, utilizes this knowledge in assisting the genealogists with decisions and strategies. The system contains an inference engine, an explanation generator, and the genealogical data. Knowledge acquisition is not automated. A natural language interface would be extremely helpful, yet is beyond the scope of this project.

CHAPTER 2 - GENEALOGICAL PROCESS

2.1 Introduction

A professional genealogist has different research needs than the hobbyist. These groups approach the research in different ways. A hobbyist approaches the genealogical process in a less rigorous, formal manner than the professional researcher. This relaxed approach has an adverse effect on the investigative strategies established, the type of data collected, the manner in which the data is organized, indexed and compiled. This chapter summarizes in section 2.2, the process an amateur genealogist goes through. In contrast, section 2.3 presents the professional's approach. Section 2.4 briefly describes the existing genealogical computer systems, followed by a conclusion in section 2.5.

2.2 The Genealogical Hobbyist

The genealogical hobbyist's primary goal is to trace his family back to a particular ancestor or generation. The process of collecting family data normally spans many years. This information must be organized as it is obtained, so that it will be accessible for analyzing family relations and establishing lineages.

The hobbyist generally has no overall investigative plan. Proceeding in a rather hit-or-miss fashion, information is gathered as it is stumbled upon. The information comes from diverse sources. Informal sources include interviews with relatives, old pictures, letters, and clippings from a relative's attic, family albums, Bibles, heirlooms, newspaper, wedding and funeral notices, and tombstones. Formal sources would include vital documents such as birth, marriage, and death certificates, court, church, and census records. The predominance of information the hobbyist collects comes from informal sources, due to the difficulty of access and time required. For example, in Kentucky and Virginia, vital records can not be found in the local county court houses. They are kept in state archives at the state capital. The records must be formally requested by mail, or a trip to the state capital must be made [MI86].

The information collected is organized using standard genealogical work sheets or charts. The ancestral, or pedigree chart, is usually the first chart to be used. Depicted in figure 2.1, this chart depicts the immediate ancestors of the person whose name appears on the single line on the left side of the page. The page reads left to right, with increasing generations on the right. The name of a given ancestor's father is on the top line of the

parent pair, with the mother's name on the bottom. The vital data, such as place and date of birth, marriage, and death is filled in as it is collected.

The line of descent chart has basically the same format as the ancestral chart, except that the lineage flows from a progenitor (on the left) to more recent descendants on the right.

Family unit, or group charts, are made for each male ancestor found on the ancestral chart. This chart, depicted in figure 2.2, assembles information about this family unit. The vital statistics of the husband and wife are given, as well as noting any previous marriages. This chart links the children to the parents. The children's vital statistics are recorded as well as identifying who they married. The source type and location should be specified for each piece of information.

The family unit charts must be linked to the ancestors on the ancestral chart. This can be accomplished by assigning codes in such a manner that each ancestor is designated by a unique number. Each ancestor's code is noted on the ancestral chart and is identified on the corresponding family unit chart, thus establishing a manual linkage system. These unique codes are also

assigned to all research notes, documents, maps, and pictures, linking these documents to the ancestors.

As the ancestral (or descendant) information is being collected, a genealogical lineage may be formulated. Figure 2.3 illustrates a typical genealogical chart. In general, the names of the descendants proceed from the earliest to the most recent. The relationships are given as well as any other pertinent facts.

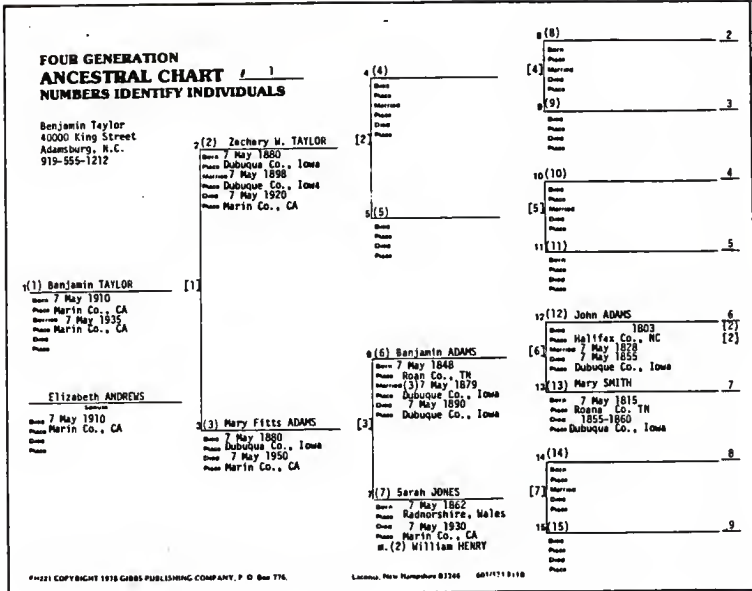


Figure 2.1 Example of an Ancestral Chart

NOTE:
 * = Direct Ancestor
 / = Line Through One Column

FAMILY UNIT CHART

PREPARED BY Benjamin Taylor DATE 30 June 1980

HUSBAND Henry ADAMS OCCUPATION "planter"; Sheriff ANCESTRAL CHART # 6(2)(2) , 6-4(48)(24)

DATE - DAY MONTH YEAR	CITY	COUNTY	STATE or COUNTRY
Born <u>bef. 1755</u>			<u>England</u>
Chancelled			<u>+45 on 1800 Census</u>
Married <u>ca. 7 May 1770</u>		<u>Halifax</u>	<u>NC</u>
Deed <u>1819</u>			<u>Halifax Co. Marriage Bond</u>
Record			<u>" " Estates Records</u>
FATHER	OTHER WIVES		
MOTHER			

DATE - DAY MONTH YEAR	CITY	COUNTY	STATE or COUNTRY
WIFE maiden name <u>Barbery JOHNSON</u>			
Born <u>abt. 1755</u>			<u>-45 on 1800 Census</u>
Chancelled			
Deed <u>ca. 1805</u>			
Record			
FATHER	OTHER HUSBANDS		
MOTHER			

SEX	CHILDREN	DATE OF BIRTH	BIRTHPLACE			DATE OF FIRST MARRIAGE		DATE OF DEATH	
			City	County	St./Ct.	Name of Spouse	City	County	State/Country
M	Henry	48.1	ca. 1771	Halifax	NC	ca. 1795	Anne Smith		
M	John	48.2	ca. 1773	"	"		Mary Johnson	1850-1860	Halifax Co., NC
M	William	(24)48.3	ca. 1775	"	"	7 May 1800	Mary Smith	7 May 1823	Halifax Co., NC
F	Anaryllis	48.4	ca. 1780	"	"	ca. 1800	John Williams		
M	Joseph	48.5	ca. 1782	"	"	did not marry		7 May 1815	Halifax Co., NC
M	James	48.6	ca. 1785	"	"	Elizabeth ?	ca. 1805		
	Mary	48.6.1							
	Joseph	48.6.2							
	David	48.6.3							

©1978 COPYRIGHT 1978 GIBBS PUBLISHING COMPANY 31 Church Street P.O. Box 776 Lenoir N.C. 27546 601 5248118

Figure 2.2 Example of a Family Group Sheet

NUMBER LETTER	NUMBER NUMBER	FINAL COMPILATION FORMAT
1	1	HENRY ADAMS
1a	1.1	+1 i Henry Adams b. ca. 1771
1b	1.2	+2 ii John Adams b. ca. 1773
1c	1.3	+3 iii William Adams b. ca. 1775
1d	1.4	+4 iv Amaryllis Adams b. ca. 1780; m. John Williams
1e	1.5	5 v Joseph Adams b. ca. 1782, d. 7 May 1815, d.a.p.
1f	1.6	+6 vi James Adams b. ca. 1785
1a	1.1	1. HENRY ² ADAMS
1a1	1.1.1	i James Adams
1a2	1.1.2	ii Amaryllis Adams
1b	1.2	2. JOHN ² ADAMS
1b1	1.2.1	+7 i John Adams
1c	1.3	3. WILLIAM ² ADAMS
1c1	1.3.1	+8 i John Adams
1c2	1.3.2	ii Henry Adams
1c3	1.3.3	iii William F. Adams
1c4	1.3.4	iv a daughter, name unknown
1d	1.4	4. AMARYLLIS ² ADAMS
1d1	1.4.1	i John Williams
1d2	1.4.2	+9 ii Henry Williams
1f	1.6	6. JAMES ² ADAMS
1f1	1.6.1	i Mary Adams
1f2	1.6.2	ii Joseph Adams
1f3	1.6.3	iii David Adams
1b1	1.2.1	7. JOHN ³ ADAMS (<i>John</i> ²)
1b1a	1.2.1.1	10 i William Adams: b. 1827, Halifax Co., NC
1b1b	1.2.1.2	11 ii John Adams: b. 1832, TN
1b1c	1.2.1.3	12 iii Mary Adams: b. 1834, TN
1b1d	1.2.1.4	13 iv Amaryllia Adams: b. 1838, TN
1b1e	1.2.1.5	14 v Henry Adams: b. 1845, TN
1b1f	1.2.1.6	15 vi Jane Adams: b. 1847, KY
1c1	1.3.1	8. JOHN ³ ADAMS (<i>William</i> ²)
1c1a	1.3.1.1	i Henry Adams
1c1b	1.3.1.2	ii Amaryllia Adams
1c1d	1.3.1.3	iii Mary Adams
1c1e	1.3.1.4	+16 iv Benjamin Adams
1d2	1.4.2	9. HENRY ³ WILLIAMS (<i>Amaryllis</i> ²)
1d2a	1.4.2.1	i Henry Williams
1d2b	1.4.2.2	ii Anne Williams
1c1e	1.3.1.4	16. BENJAMIN ⁴ ADAMS (<i>John</i> ³ <i>William</i> ²)
1c1e1	1.3.1.4.1	i Benjamin Adams
1c1e2	1.3.1.4.2	+17 ii Mary Fitta Adams m. Zachary W. Taylor
1c1e2	1.3.1.4.2	17. MARY FITTS ⁵ ADAMS (<i>Benjamin</i> ⁴ <i>William</i> ³ <i>John</i> ²)
1c1e2a	1.3.1.4.2.1	+18 i Benjamin Taylor
1c1e2a	1.3.1.4.2.1	18. BENJAMIN ⁶ TAYLOR (<i>Mary Fitta</i> ⁵ <i>Benjamin</i> ⁴ <i>John</i> ³ <i>William</i> ²)

Figure 2.3 Example of a Genealogy Lineage Chart

The end result for many hobbyist's is a collection of ancestral and family group charts, family pictures, and stories, invariably being distributed at family reunions.

The three major problems that plague genealogical hobbyists are incorrect ancestries, lack of organization, and inadequate numbering schemes.

Incorrect ancestries can result from a variety of reasons. Many of the informal sources that are used are not verified or proved to be accurate. Thus, the conclusions made based on these facts will not be correct. Also, in order to correctly interpret the meaning of a source, an understanding of the laws, customs, and history of the particular time and locality of the event is required.

A lack of organization of the text, documents, and charts will 'bury' the genealogist. Furthermore, without an organized way to index files, facts that may provide important clues may never be uncovered.

The Achilles' heel for the home genealogist is the manual numbering scheme required to link the data. A scheme which is adequate today, may not be adequate three years from now, requiring a massive reorganization of the files.

2.3 The Professional Genealogist

2.3.1 The Professional Approach

As stated in chapter 1, professional genealogy involves comprehensive and exhaustive research. The professional must assure that the finished product, which is indeed the product of a business, is an accurate genealogical lineage and historical chronicle. The proof of the lineage is obtained from facts and inferences derived from public records and documents, as well as from unofficial sources. General strategies for the use of each of these documents can be established, assisting in the accumulation of 'leads' or clues.

The basic genealogical process proceeds from the known to the unknown. Research starts with an exhaustive and accurate description of the current generation. Each generation's information is assembled before moving on to the next. At each step, an analysis must be made on what has been learned up to this point. Strategies are established to find what is still not known. The key to genealogy is the discovery of the clues which lead directly, or indirectly to the piece of information or proof that is needed.

The following sections present a brief overview of the major sources of information a genealogist searches, and a brief sample of the strategies associated with each. This information is derived from the project's contributing genealogical expert, Helen F.M. Leary [HL86,AL80]. The strategies are based on North Carolina's genealogical guidelines and records, yet, the principles would apply elsewhere.

For information on a particular ancestor, the suggested search order of the public records is: 1) vital records, 2) censuses, 3) wills and estate papers, 4) deeds, 5) records of collateral surnames, and 6) secondary sources.

To search these records, a genealogist must have an approximate idea of the location and time of the event recorded. This approximation can be inferred from an analysis of the data known up to this point.

Information in one record group will give clues in the location of information in other record groups. The above steps should be repeated for each new generation being researched.

2.3.2 Vital Records

Vital records include information on birth, death, marriage, and divorce events. Bibles, newspapers, and

tombstones can also be used to discover birth, marriage, and death information.

It is not unusual to find discrepancies in information such as date of birth. The order of precedence for conflicting information is:

1. date of birth and names of parents from birth certificate over this information from death certificate, tombstone, or Bible;
2. date of death from death certificate over tombstone or Bible; and
3. Bible date and name over tombstone inscription when Bible entry was made at the time of the event.

To illustrate how information can be derived from vital records, a brief discussion of marriage records follows.

Marriage records consist of marriage bonds, licenses and certificates. Marriage bonds record marriages in North Carolina from 1741 to 1868, and are the only public marriage record prior to 1851. The bond was required of persons intending to get married. If the marriage took place, a marriage license was issued. This license was not required by law to be recorded. It is highly probable that a portion of the bonds recorded are for marriages

that never took place. Used in conjunction with other records, though, this information can be quite useful.

The bondsman and witnesses on the bond should be noted. Often these people were relatives and friends. These names could lead to other records that provide helpful information in determining who this ancestor is. The clergyman's denomination on the bond could be a clue to the location of a particular church's records.

The marriage information, as with all vital records, should be compared with information from the other records, in order to discover new clues. For instance, by comparing marriage information with wills, and estate records, the following types of information can be deduced: the approximate age of children, the parents of grandchildren named in a will, the locality of a missing son (he may be living on land inherited by his wife), or possibly the new surname of the deceased ancestor's remarried wife.

2.3.3 Census Records

Censuses are federal records compiled, periodically numbering people. It provides an official list of persons residing in each household, including age, birthplace, occupation, marital status, etc.. The information is not

always accurate. Ages in the 1850 - 1900 censuses can vary up to twenty years. A search and comparison of multiple censuses should be undertaken for the years in which the ancestor appeared.

The census, especially when used in conjunction with other censuses and records, is a vital source of information. Second or third marriages may be indicated by the gaps in the ages of the children, difference in the age of the wife versus the age of the husband, or the age of the wife at the birth of the oldest child in the household. A deceased spouse may be indicated on the census's mortality schedule. Grandparents or in-laws can be inferred. Migration patterns of the family can be determined. The dates a male reached 16 - 23 can lead to possible tax records. The age for military service can lead to military records.

2.3.4 Wills

The will is a document which declares a person's wishes regarding the disposition of his belongings after his death. To begin a search of wills, the state's devisee (heir) index is checked to determine if the ancestor being searched for is listed in someone's will. If not found in this index, all wills for his surname in the county he

lived must be searched. If the time frame in which he lived is not known, the search could take forever.

A general strategy to shorten this process is:

1. The ancestor's father was alive at least 9 month's prior to the birth of the ancestor.
2. The following formula calculates the latest he could have died: subtract 20 years from the ancestor's birth date to arrive at his father's birth date, add a proposed life span of 70 years, and add another 10 years for good measure.

This strategy derives a time bracket in which to search all wills matching the ancestor's surname.

Religious affiliation may be indicated from the will. A Quaker ancestor probably would have an absence of the "in the Name of God" clause. The information in the will should be compared with information derived from censuses and other records. If the family mentioned in the will matches the census enumeration, then unless your ancestor is mentioned in the will, this is not his parent. If the census number is greater than the number in the will, it's possible that your ancestor was left out of the will.

Estate papers, court proceedings against the will, and deeds can provide additional information.

2.3.5 Estate Records

There are a diverse type of estate records. A few types include bonds, inventories, bills, receipts, and divisions. The majority of estate records are stored in the boxes of county records. The same search time period that was used for searching wills must be used for the estate records. An analysis of estate records can give clues about the deceased or his family, such as, ownership of business or land, public offices held, occupation, and the economic condition of the time. The estate division may provide approximate sexes, ages, and married surnames of heirs.

2.3.6 Land Records

All land that the ancestor bought or sold can give clues needed for our investigation. Land records can give us clues to the age of the grantor or grantee. In North Carolina a man couldn't sell land until he was 21, yet could buy or be granted land before he was of age. Land deeds can indicate how the land was obtained, the reason for selling or granting it, and the condition of the people involved. It is noted, that the man who signs a deed with a signature mark is not the man who signs a will with his name. Conversely, the man who signs a deed with

his name, may or may not be the man who signs a will with a mark.

2.3.7 Tax and Court Records

A tax list presents the tax responsibilities of people. Men had to pay an additional poll (head) tax for family members in his household. It is possible, using tax lists, to determine which household members listed in the census were not family members. Tax records, when used in conjunction with other types of records can be very useful.

The analysis of all of the previous mentioned records and a determination of the law at the time will govern how court records are used. They must be interpreted in light of the time in which they were recorded.

2.3.8 Professional Genealogist Summary

The overview presented above, barely touches the surface of genealogical research techniques and strategies. It is given to illustrate the magnitude of the types of decisions and inferences that are made. It takes years, and years of diligent study and experience to accumulate the genealogical knowledge needed for efficient, and accurate research. It is clear that an expert genealogical system that is flexible enough to be tailored

to their locality, would greatly increase their research productivity and serve as an excellent training tool.

2.4 Current Genealogical Systems

Currently there are over fifty commercial genealogical systems on the market. These systems generally all produce pedigree charts and family group records. Beyond that they all differ greatly in function, capacity, and capability. The systems all run on various personal computers, ranging from IBM compatibles to KAYPRO's. A brief description of the four most popular systems is given.

TREESEARCH, by Array Systems, allows the user to enter research data on a portable Epson HX-20 lap top computer. Data is later dumped into a desk top computer and merged with existing files. It allows layered searches and sorts. A telecommunication interface allows the user to link to another personal computer. It has no soundex capability. Soundex is an indexing system used by the National Archives for some censuses, in which use of a numerical code facilitates the identification of names appearing under variant spellings. Descendant charts are not produced. The largest drawback is it's limited storage capability.

PATRIARCH I, by Cyclone Software, will only run on an Apple II computer. It has fifty user defined fields in its program, allowing different genealogical needs to be facilitated (i.e. Jewish genealogy). It will not produce pedigree charts, and does not have soundex capability. It has no telecommunication facility. It has a particularly nice searching and indexing function. It's largest problem is that it will only allow short, fixed length text records to be entered.

FAMILY ROOTS, by Quinsept, runs on a variety of computers. It has been on the market the longest, providing a whole package of associated report type packages. It is one of the few systems to have a soundex capability. It's greatest drawback is that it can not produce research data indices or sort source material.

ROOTS II, by Commssoft, also provides a soundex capability. It will only run on IBM compatibles. It has very efficient disk storage. It will store and retrieve pictures, maps, plot plans, etc.. It does not have a telecommunication capability. Of the currently marketed systems, this is the most sophisticated of the group.

All of the above systems are tailored to the needs of the amateur genealogist. None of the systems provide for an integration of text with the databases. The primary

result of these systems is the generation of genealogy charts. The professional genealogist's final product is a text. The major drawback of the above systems is the lack of intelligence. They provide no relationship checking, no inference capabilities, and no means to store, manipulate, and distribute knowledge.

2.5 Conclusion

The professional genealogist's needs are different than that of the home hobbyist. The professional genealogist needs a tool that assists in setting up research strategies, gives suggestions, makes inferences, and can be used as a learning tool. The existing genealogical computer systems, while providing the home hobbyist an adequate tool, falls short in meeting the sophisticated needs of the professional.

CHAPTER 3 - REQUIREMENTS

3.1 Introduction

Professional genealogists need a single, integrated system that assists them in their research and publishing endeavors. This system should provide the storage, linkage, and manipulation of research data. It should have a text processor with the capability to interface with the structured genealogical database. It needs to be easy to use and to understand. Lastly, it should provide assistance in their analysis and decision process. The system should consist of a user front-end and three sub-systems: an expert system, a text processor, and a database management system.

The objective of this thesis is to address the integration of expert system technology into a genealogical information system. Although the user front-end will be discussed, the main focus of this design will be on the expert system. The text and database components will be the design effort of another Kansas State University thesis [BE88]. The two designs are tightly integrated into a total information system. The relationship of these components is illustrated in Figure 3.1.

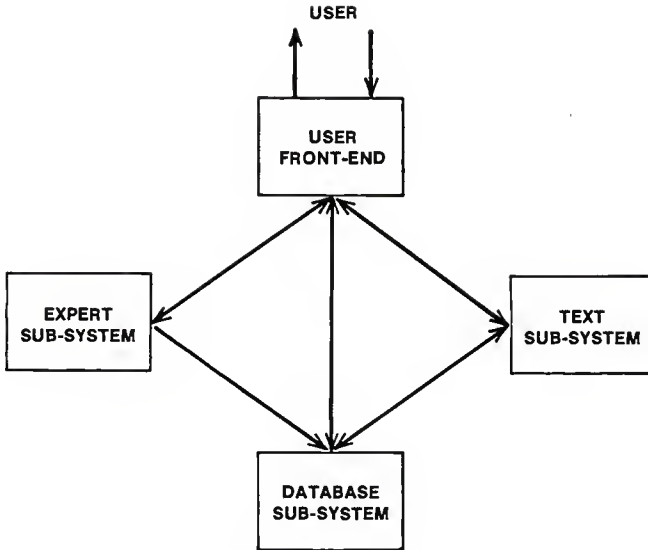


FIGURE 3.1 PROFESSIONAL GENEALOGICAL INFORMATION SYSTEM

3.2 General Requirements

Most professional genealogists have small businesses, working out of their homes. The genealogical information system must be designed to run on an affordable, personal computer. The computer should have a large memory capacity. A printer is required for printing text and genealogical charts.

The typical genealogist is not knowledgeable about computers. The system must communicate with the user in a clear, non-technical language. The system must require a minimal amount of effort on the part of the user. It must be designed with soft-fail features, providing informative diagnostics. A complete, yet easy to read user's manual must be provided.

The genealogical information system must be able to store and process massive amounts of research data. The user must have the capability to add, delete, or modify the data. The system must provide a versatile query function, allowing the user to make standard and ad hoc query requests to the information in the database. Linkage and relationships between the data must be facilitated.

The genealogist requires a text processing environment in which to type research notes and to prepare the

professional genealogist's product, a published text. The capability to dynamically insert database information into the text must be provided.

The information system will have the capability to store and utilize genealogical knowledge that is needed for analysis and decision making. It must have the flexibility of adding or deleting this knowledge. The user must be able to start a dialogue, and ask for an analysis or hypothesis based on the genealogical data and knowledge stored in the system. The capability to temporarily modify the decision making process must be provided. A complete explanation of each derived decision must be available upon user request.

3.3 Specific Requirements

This section addresses the specific requirements of the components of the genealogical information system, the user front-end and the expert system. Section 3.3.1 presents an overview of the user front-end component and the capabilities that are required. This section is followed by a discussion of the expert system requirements.

3.3.1 The User Front-end

The user front-end component will provide the interface between the user and the other sub-systems of the genealogical information system. It will communicate through a combination of easy to understand menus and problem oriented dialogue. The grammar and syntax of the user's commands will be parsed and interpreted. Error messages will be returned for incorrect commands, along with a prompt for the user to re-enter the command. The user front-end will interpret correct commands and pass control to the specified sub-system. All interactive output from the three sub-systems is formatted and presented back to the user.

3.3.2 The Expert System Component

The expert system component will provide intelligence to the genealogical information system. It will facilitate the distribution of knowledge from genealogical experts. The expert system will store a high level representation of the genealogist's rules-of-thumb, strategies, and general genealogical knowledge. This knowledge, when customized, will include localized information, such as, county, state, and federal laws governing the use of public records.

The contributing professional genealogist is the expert who will be supplying the knowledge that the system uses. The system will initially be loaded with the rules derived with the aid of the expert. The design will accommodate continual change. The addition of rules will allow the system to be customized to each specific professional genealogist who would use it.

The genealogical knowledge will be represented in the form of rules and stored in a knowledge base. These rules are structured into packets, that contain related rules. The user will have the capability to add, modify, or delete rules from the system. As this change occurs, the rule base will be re-organized in order to retain the relationships and semantics of the stored knowledge.

The expert system will incorporate search strategies to narrow down the possible solution paths and the set of rules to be used. Forward or backward chaining will be used to arrive at a final solution.

The expert system will be activated in three different ways. The system will be activated when the user requests a dialogue with the expert system. It will also be activated when the user requests to add, delete, or modify the rule base. Lastly, it will be activated if the

database system requests a decision or analysis of database information.

Through a dialogue, facilitated by the user front-end, the expert system will allow the user to hypothesize on unsupported facts or request an analysis of existing data. The genealogical procedural knowledge, in conjunction with the information in the database, will be used for the analysis. Suggestions, decisions or conclusions will be returned, as requested.

The user, by choosing the expert system maintenance option, will be able to selectively display the rules in the knowledge base. The capability to add, delete, or modify these rules will be provided. The system must provide for the integrity and security of the knowledge base. An audit record containing the genealogist's name, and the date will be created for each rule added. Only the genealogist who added a rule will be allowed to delete or update it.

The user will also have the option to request a complete explanation of how a decision or solution was derived. This includes a list of all rules that were used. The user will have the capability to modify the decision process by "backing up" to a particular rule listed and

temporarily over-riding the rule's conclusion. This provides for a modification of the decision process.

The expert system will communicate with the database system. This database system will send requests for data analysis and decisions to the expert system, which will return the results. These decisions will be used by the database system in order to properly establish linkages. These linkages are necessary in order to add, delete, modify, or query the data.

CHAPTER 4 - DESIGN

4.1 Introduction

This design chapter addresses the integration of expert system and database technology. The genealogical information system design is the result of this effort. This chapter concentrates on the incorporation of intelligence into a database processing environment. 'Experts' are established to provide decisions and intelligence to the information system on the genealogical research data provided by the user.

Section 4.2 will provide a high level overview of the information system. This paper addresses only a portion of the total system design. The text and database functions are the topics of another Kansas State University project [BE88]. Section 4.3 discusses the areas of the above design in which this paper will address. The areas that are beyond the scope of this design will be noted. Section 4.4 describes the concept of experts and how they relate to this subject area. Section 4.5 addresses the interaction between the database control sub-system and the experts. A high level example of an expert is presented in section 4.6, followed by a summary in section 4.7.

4.2 Overview of the Genealogical Information System

4.2.1 Introduction

The genealogical information system is designed to meet the needs of the professional genealogist. These needs include an easy to understand and to use interface, a means to store and manipulate data on people and sources, a facility to enter additional unstructured text information and assistance in analyzing the data collected.

The information system is composed of a user front-end which interacts with the database, text, and expert sub-systems. The relationship of these sub-systems is depicted in Figure 4.1.

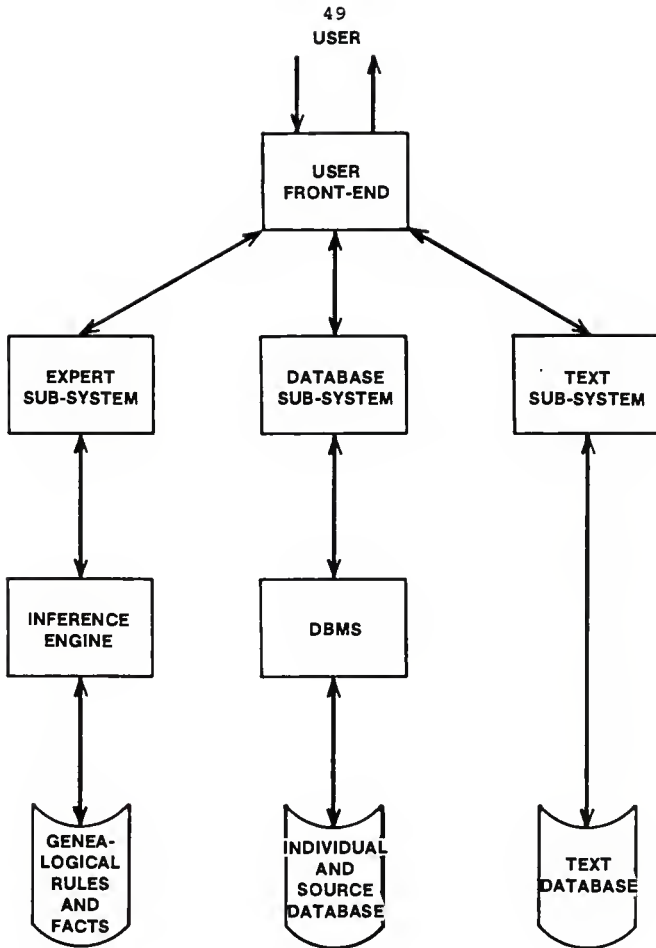


FIGURE 4.1 RELATIONSHIP OF THE GENEALOGICAL INFORMATION SYSTEM COMPONENTS

The following overview is a brief description of the entire system. During the design process, the extreme complexities and intricacies inherent in genealogical data were discovered. Some of the functions of the system were found to be beyond the scope of this thesis. The problems did provide, though, an interesting insight into incorporating expert system technology into large, complex problem domains. These difficulties are discussed in detail in Chapter 5. A functional overview of the information system and its components is presented. The unaddressed features will be referenced for future study. Section 4.3 will discuss the portions of the system that this design will address.

4.2.2. System Overview

The **user front-end** is the sole interface with the user. Communication is established through the use of screen templates which provide easy to understand menus and screens. The primary menu gives the user a choice of a database, a text, an expert system utility, or dialogue function.

In addition to providing menus, the front-end interacts with the user in a problem-oriented, stylized English language. The user's commands and requests are parsed and interpreted. If a request or command is invalid, because

of either incorrect syntax or semantics, the front-end provides an error message. The user is requested to re-enter the command. Correct requests are interpreted. Control is passed to the sub-system whose functions were specified. A screen template is presented for the chosen function. For example, the database control subsystem template would display a menu of options: adding, modifying, deleting, and querying data. In addition to generating all screens, the user front-end also formats all interactive output returned to the user. This output is passed to the user front-end from the three sub-systems. Figure 4.2 depicts an overview of the user front-end.

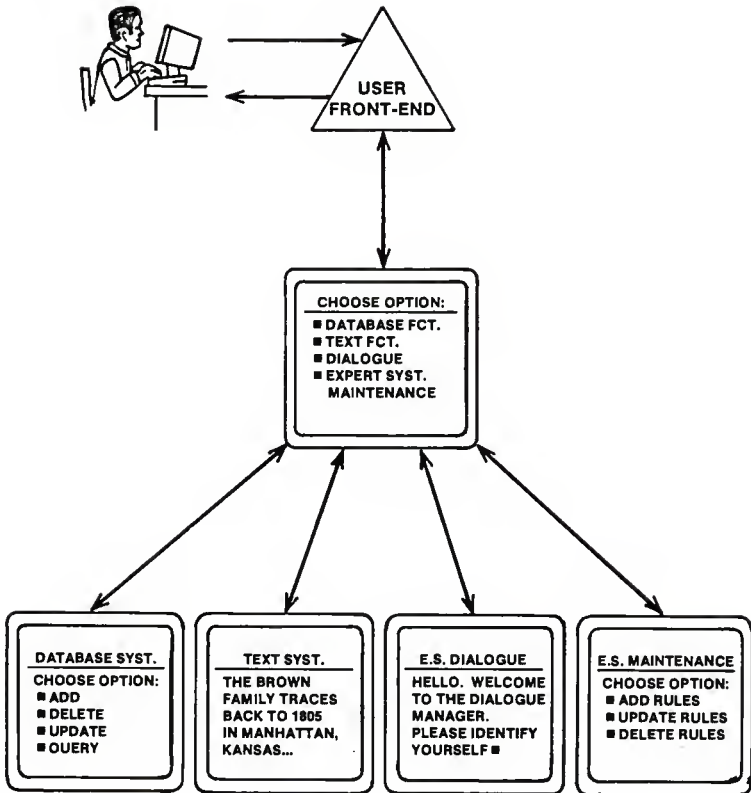


FIGURE 4.2 OVERVIEW OF USER FRONT-END

The database control system is activated when the user requests to add, delete, modify, or query data in the genealogical database. All information in this system is derived from source records (birth certificates, wills, land deeds, etc.), that the user adds to the database. Each source record entered provides information about a primary individual and possibly secondary individuals. For example, a birth certificate provides information about the individual born and his parents. The source document can also provide an abundance of other information, such as, dates, addresses, important events, characteristics, occupations, military service, and cause of death. The source document may also contain information about the document itself. Relationships to other individuals is often given or inferred from the source document. The database contains information about the individual, the people related to the individual, and the source records providing this data.

In genealogical research, the source records collected often provide incomplete information. This provides for complex relationships (linkages) between individuals and between individuals and source data. For example, several source records may describe a Tom Jones. Until sufficient proof is established that they are the same person, they are treated as separate individuals. The database sub-

system interfaces with the expert sub-system to obtain notification of the existence of this proof. The expert system provides this analysis.

The data items on the source documents must be analyzed before the data is added to an individual's record. The data items on the source record include all fields except for names. Names infer identities and relationships. They are analyzed separately. The precedence, integrity, and validity of data items must be determined before the source record information is added. Expert system modules provide this intelligence to the database sub-system. Decisions must also be made when these data items are modified or deleted. For security measures, the identification and date the data item was added is stored. It is important to identify the research that each genealogist using the database is doing. If one genealogist is allowed to delete a piece of information critical to another's research, the integrity of the data may be compromised. For this reason, only the genealogist who entered the data is allowed to update or delete the data. When deleting data, decisions about the precedence of the remaining items must be made. The expert system modules, called experts, are activated when data is entered. They interact with the database sub-system,

requesting database tuples and providing decisions in return.

When the user requests to enter a source record, a screen template is presented which contains fields for each data type in the source record. The user enters the source record data items that are to be added. This record is moved to a frame in memory which is accessible to the sub-systems. The population of the slot in the frame in conjunction with the user specifying with the add specification, triggers an expert sub-system expert. The expert analyzes the data item in relation to the data values of the attribute in question. Communication occurs interactively between the experts and the database sub-system.

The text sub-system is given control when the user requests to enter, modify, or delete text. This sub-system provides standard text editing features, such as scrolling, inserting, deleting, etc.. The user may dynamically integrate database information into the contents of the text. The text sub-system interacts with the database control sub-system to determine the data fields to be inserted into the text. If decisions are required, such as which Tom Smith's data should be

returned, the database sub-system communicates with the expert sub-system.

The expert sub-system is passed control from the main menu when the user requests a dialogue session or a maintenance option (adding or deleting genealogical rules or facts). A dialogue session allows the user to request, in stylized, structured English commands, via the user front-end, an analysis of relationships, identities, and strategies. The user is given the ability to hypothesize. The user presents the sub-system with an unsupported fact and requests an analysis of the impact this fact may have on the existing data. For example, a user may want to know, "Fred Brown's will named Joe Brown as an heir. Can we assume that Joe Brown is Fred's son?" A conclusion is returned with an explanation of how it was arrived.

The expert system uses a knowledge base of genealogical facts and rules in conjunction with the genealogy database to derive conclusions. A small prototype subset of facts and rules was provided by the contributing professional genealogy expert. The genealogical knowledge in this design is grouped into units called experts. Each expert is a modular unit of logic, which provides a decision or analysis of a specific problem. The experts interact with the database sub-system. They provide decisions on when

and where data should be linked. The database sub-system requests decisions on relationships and identities. The expert sub-system performs an analysis and presents the results to the user. The user is required to make the final choice. The user's decision is returned to the expert sub-system and is passed to the database sub-system.

The users are allowed at any time to request an explanation of all solutions or suggestions. This explanation provides an audit trail of the rules that were used to derive the solution. Figure 4.3 depicts the relationships between the data experts and the database sub-system.

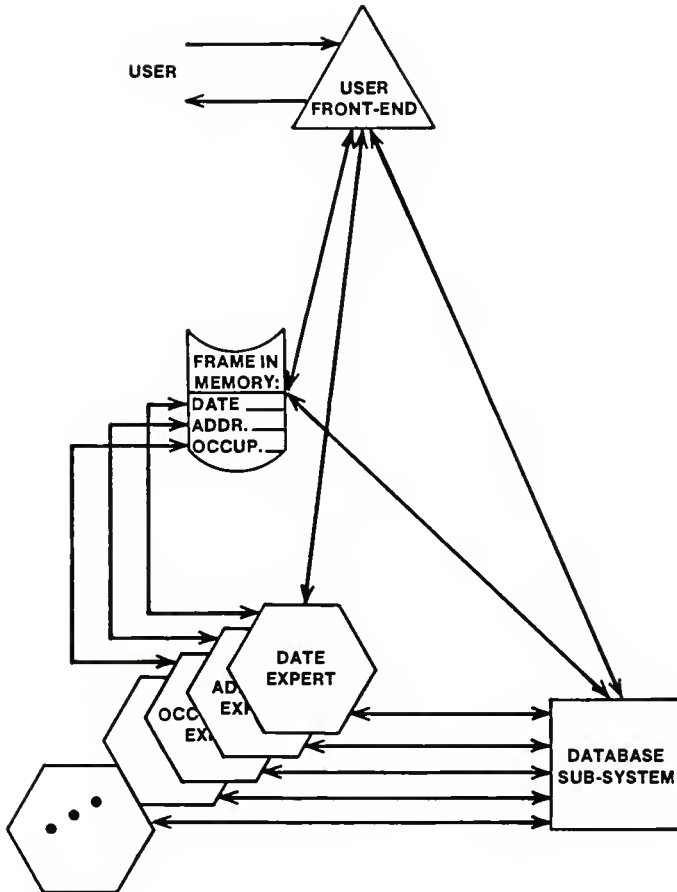


FIGURE 4.3 RELATIONSHIP BETWEEN DATA EXPERTS AND THE DATABASE SUB-SYSTEM

4.3 Scope of this Design

Initially, the scope of this thesis was to design an expert system which completely aided the genealogist in all of their decision making processes. This was to have included interactive dialogue sessions with the user, the ability to create new rule bases by adding, modifying or deleting rules, the ability to temporarily over-ride these rules, an explanation generator, and an efficient inference engine employing a well chosen search strategy. Due to problems that arose during the original design process, the scope of this thesis has been modified. Chapter 5 contains a detailed description of the problems and insights gained from early design efforts.

The scope of this thesis will be to address the concept of employing expert system technology, in the form of experts, in a database information system. These experts provide dynamic intelligence to the database sub-system. An analysis is performed on every data type found in the source document. This paper will not address the design of additional experts that ideally would provide search strategies, decisions on relationships, and identities. Reference will be made to these additional experts in the overall flow of the system.

4.4 Experts: Knowledgeable Modules

4.4.1 Background

At the start of this design effort, it was this author's position that genealogy provided a problem area that was suitable for expert system technology. It exhibited a need for decision-making assistance and for a tool to distribute hard to obtain knowledge. Many of the "possible, justified, and appropriate" criteria established by Donald A. Waterman [DW86] were met.

After the design process began, it was determined that current, commonly used expert system approaches did not address the difficulties that arose (refer to Chapter 5). It was discovered that genealogical research is very complex. It is full of intricate and subtle relationships and inferences. Establishing a relational database to store this data and to also retain the relationships was difficult [BE88]. The decisions that genealogical experts have to make, using this data, require not only a knowledge of rules and strategies, but also an intuitive feel for the inter-relationships of the various data items.

An attempt was made to create a subset of expert system rules from the knowledge extracted from the contributing

genealogical expert. This set, which provided strategies, precedence, rules of thumb, and general genealogical knowledge, was extremely complex. The rules were very inter-related. The design problems created by the complexity of the rules were:

1. The intricate inter-relationships of the rules resulted in a loss of semantics.
2. The loss of modularity made it extremely difficult to design a way for rules to be added or deleted from the knowledge base.
3. The loss of modularity also made it difficult to devise a method to allow the user to temporarily override the decision process.
4. The large mass of inter-related rules provided many possible decision routes (using either forward or backward chaining), which made the decision process inefficient.

From this early design effort, it was clear that if expert system technology was to be applied to a large, complex knowledge base, a different approach was necessary. From this, the concept of experts arose.

4.4.2 What is an Expert?

An expert is a modular, independent unit which contains knowledge derived from an expert in a specific problem domain. The expert acts like a function on data. The rules and knowledge are applied to the data to derive a decision or analysis. Because an expert is modular and independent, other experts can be added or deleted from the expert system without distorting the semantics of the knowledge represented. The experts are designed to solve specific not general problems. The efficiency problem of multiple solution paths disappears.

In the genealogical information system, there is an expert for each decision that the genealogist needs to make. Decisions must be made on each data item found in the source document before the data item is linked to an individual's record. Other decisions include the identification of individuals, the determination of relationships, and the derivation of strategies. This design will address only the decisions on a datum. The other decisions are beyond the scope of this thesis.

4.4.3 Decisions on a Datum

An expert has been established for every function and every data type. The expert acts as a function on the data, returning a genealogical decision.

These data experts are designed as demons. Demons are procedures that are activated by the presence of a piece of data in a field. The user enters the contents of a source record into the screen template. This information is placed in a corresponding frame in memory with a slot for each data item. The presence of a particular data item, in conjunction with the type of operation being performed, i.e., add, delete, or update, determines the demon which is to be 'triggered.'

The expert is self-contained. In order to make a decision on a data item, depending on the rules being applied, information from the database may be needed. The expert has the capability to interact with the database sub-system. Communication between the two sub-systems establishes the transferal of database information and derived solutions or decisions. The database sub-system requires these decisions in order to properly link data values to database tuples. This interaction is described in more detail in Section 4.5. Once an expert has

completed it's function, it disappears until it is activated again.

4.4.4 Types of Experts

There is an expert associated with each of the major functions performed upon each data type. For example, for the birth certificate birthdate, there is an add expert, a delete expert, and an update expert. These experts are established to preserve the precedence, integrity, and validity of that date field.

Add experts determine if there is an existing data value for the attribute in question. If there is, a decision must be made as to which data item receives precedence. Depending on the data type, decisions are made concerning the validity or accuracy of the item.

Delete and update experts provide security to the data values. The name of the genealogist who entered the data is stored, along with the date of entry. These experts restrict the deletion or modification of data fields to the genealogist who entered the data. Security violations are recorded in a security log. Depending on the data type, other decisions are also made.

In genealogy, there are numerous types of source documents. Chapter 2 provided a detailed discussion of

the most prevalent records. A sample of the types of fields found on these records is presented in Table 4.1. Each of these fields would have an add, update, and delete expert associated with it. A functional hierarchy of these experts is presented in Figure 4.4.

The experts themselves may form a hierarchy of experts. Pieces of logic that are common to two or more experts are broken out into specific, single purpose sub-experts. For example, the add-birth-certificate-birthdate expert and the add-bible-birthdate expert both need to validate the date and determine the precedence of the existing birthdate. This modularization allows for rule changes without requiring a major modification to a number of experts. Figure 4.5 depicts an example of the hierarchy proposed.

Sample of Source Record Fields

Birth Certificate:

date of birth
 place of birth
 doctor who delivered
 weight at birth
 sex

Death Certificate:

date of death
 place of death
 cause of death
 occupation
 age
 sex
 marital status

Marriage Certificate:

date of marriage
 place of marriage
 denomination of minister
 age of wife
 age of husband
 signature mark of wife
 signature mark of husband

Census:

number of people in
 household
 place of residence
 number of white males
 number of white
 females
 number of males
 between age ranges

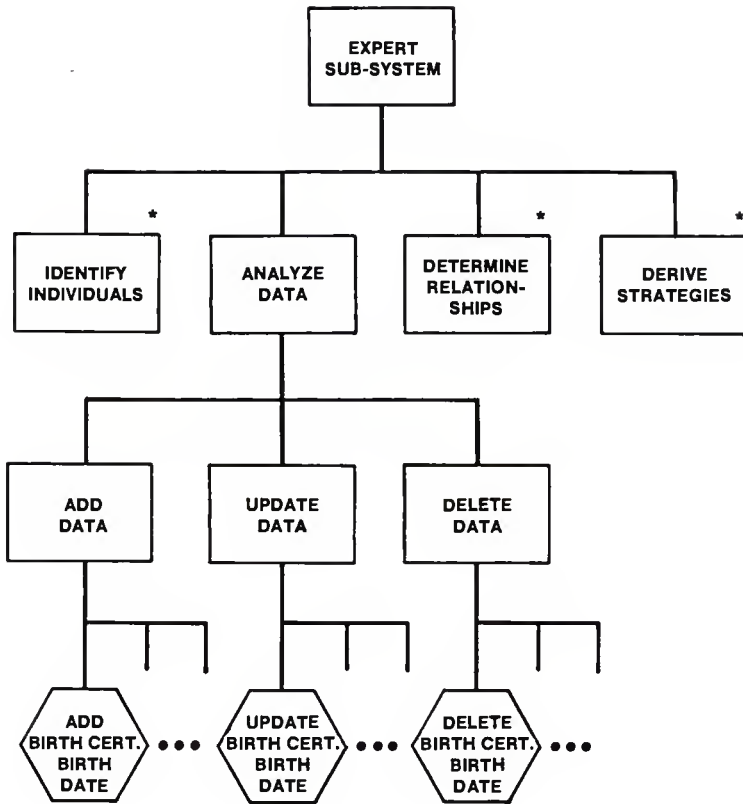
Will:

testator's place of residence
 date of will
 signature/mark
 contents of estate

Land Record:

date of deed
 place of filing
 place of land
 place of filer
 conditions of the
 deed

Table 4.1 Table of Sample Source Record Fields



* NOT COVERED IN THIS DESIGN

FIGURE 4.4 FUNCTIONAL HIERARCHY OF THE EXPERT SUB-SYSTEM

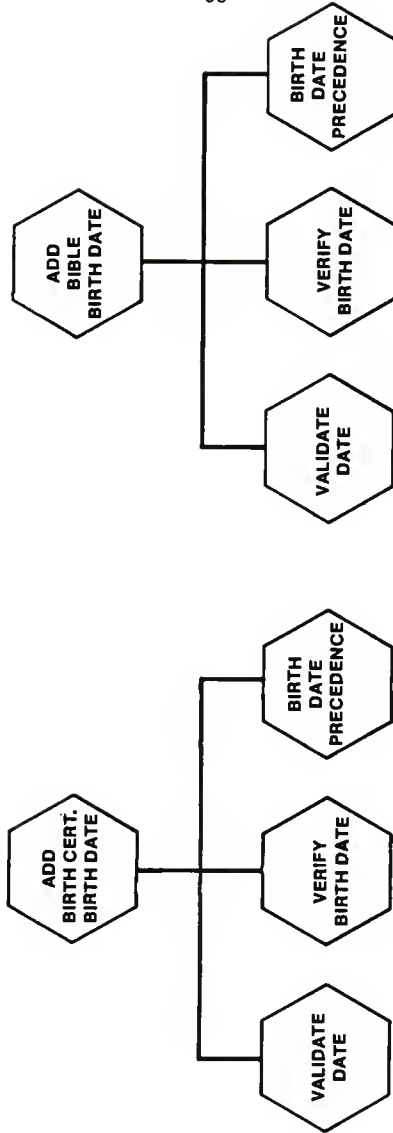


FIGURE 4.5 INTERNAL EXPERT HIERARCHY

4.5 The Expert/Database Interface

The interface between the experts and the database sub-system facilitates the interactive transferal of information in the genealogical system. This interactive communication is instigated by both sub-systems. Common interactions would include the expert sub-system requesting needed tuples, or the database sub-system requesting a decision or analysis. This section provides a discussion of the ways in which this interaction occurs. The remainder of this section presents a typical expert/database scenario.

4.5.1 Data Experts

The data experts are triggered when the source data items are placed in the frame in memory, in conjunction with a specification of the add, update, or delete function. These data items include all of the information on the source record, except for the names of individuals.

4.5.1.1 Adding Data Items

When the user enters a specified source data item and requests this item to be added to the database, the add expert for this data item is 'fired.' The add expert validates, verifies, determines precedence, and instructs the database on how to add (link) this data item.

The expert first validates the data item. It checks to see if the data item is in the correct form, i.e., numeric versus alphanumeric, and is semantically correct. For example, 35/62/90 is not a semantically correct date. If the field is not valid, an error message is returned to the user, via the front-end processor. The top two lines of each screen template is reserved for interactive communication between the user and the system. The error message from the add expert is printed in this field. The field in question is highlighted. The user is requested to re-enter the field or designate the field to be empty. When the send/receive button is pushed, the expert evaluates the new data item. It must be noted, that a source record may have certain fields that must be populated for the source record to be considered. For these source types, the corresponding experts verify that the fields have appropriate data.

If the data item entered is successfully validated, the expert checks the database tuple to ascertain if there is an existing data value in the attribute in question. This existence indicates that there is one or more data items associated with this field. In the database, the data item with the highest precedence is marked. In order to determine if the new data item has precedence, the add

expert requests from the database sub-system, information on the data items associated with this attribute. This information is composed of the data items and the sources that they are associated with. For example, for one individual, there may be three birthdates, from a tombstone, bible, and birth certificate. The add expert using the appropriate rules of precedence for this data type, determines whether the new data item or the existing marked data value has precedence. The database sub-system is instructed to mark the data value that was found to have the highest precedence.

An ownership record is created for each data item added. This record contains the name of the genealogist who added the data and the date the data was added. The genealogist is required to identify herself (himself) when entering the genealogical information system. This ownership record is utilized by update and delete commands to establish security.

4.5.1.2 Updating Data Items

The user requests to update a source record. All of the associated information for this source record in the database is displayed on the screen template. The user indicates the field(s) that is to be modified and types in the new field value. When the data item is placed in the

corresponding frame in memory, the update expert for this field is triggered.

The update expert compares the name of the genealogist who signed on with the name of the genealogist who has entered this data item. If the names are not the same, a security violation message is returned to the user. The update attempt is aborted. A security violation record is written to a security log that contains the record identification, the date of the violation, the data's owner, and the name of the violator. This log can be reviewed to monitor the security of the database.

The update expert is similar to the add expert, in that it must validate the data field entered, as well as apply all applicable verification rules. The expert system asks the database sub-system if there is an existing data value(s) for the attribute in question. The expert system instructs the data sub-system to determine if the source data item to be updated exists. If the data item does exist, then the data value is replaced with the new value. The expert requests information about any other associated data values and reassesses the order of precedence. The database sub-system is advised of the precedence decision and acts accordingly. If the data item to be updated does not exist, the field is highlighted and an error message

is returned to user, requesting the user to re-enter the field in question.

4.5.1.3 Deleting Data Items

The user can request to delete a data item by selecting the delete option from the database control screen. The specified database record is displayed. The user indicates the field(s) that are to be deleted, triggering the delete expert for this data item.

The delete expert verifies the ownership of the data item before authorizing the database sub-system to delete the field. For unauthorized delete attempts, the field in question is highlighted, and a security violation message is returned to the top of the user's screen template. The delete attempt is aborted. A security violation record is written to the security log, as described in the section on adding data items.

Once the ownership is verified, the update expert requests for the database sub-system to delete the field and to present all remaining associated data value(s). The expert analyzes this information. The database sub-system is directed on how to reestablish the precedence of this attribute.

4.5.2 Establishing Identities and Determining Relationships

The database sub-system is dependent on decisions provided by experts, in order to properly establish linkages. These decisions establish identities and determine relationships. For example, if a death certificate is entered, information from the will should be linked to the individual in the database whose certificate this is. If there are several individuals in the database with the same name, an expert is called to determine which individual is the deceased. The expert requests the needed database tuples, analyzes existing data, and returns the analysis back to the user.

4.5.3 A Sample Scenario

Genealogist, Betsy Thomas, starts up the genealogical information system. The system requests her to identify herself. The primary menu screen template is presented. Betsy wishes to add a marriage certificate. She chooses the database sub-system. She requests to add the certificate. A screen template of a marriage certificate is displayed. The contents of the marriage certificate are copied into the screen fields. Betsy presses send/receive button. The data in the screen template is moved to a frame in memory.

The database sub-system is activated by the user's request to add a record. The primary name fields (the husband and wife whose marriage certificate this is) and secondary name fields (witnesses, the minister, etc.) are determined. If the names are not already in the database they will be added. In order to determine if the individual on the certificate is the same as the individual in the database, the database sub-system calls the identity expert. This expert analyzes existing data in conjunction with the new record. The dialogue between the expert and the database sub-system goes back and forth. The expert requests information it needs from the database and the database system returns this information. After all of the data is evaluated, the expert notifies the database system of the possible identities of the individuals. The database system requires this knowledge in order to establish linkages and add the marriage certificate data to the database.

Once the identities have been established, the corresponding add expert is released for each data item entered into the frame in memory. For the marriage certificate, the experts released would include: an add-marriage-date expert, an add-place-of-marriage expert, and add-denomination-of-minister expert. These experts go off

and determine the validity and relevance and precedence of the data types.

Betsy is not a good typist. She enters a blank field for the marriage certificate date and 3333333 for the place of marriage. The respective experts present her with appropriate error messages, highlight the fields in error, and request the fields in error to be re-entered. Betsy retypes the fields that were highlighted. This time Betsy types in 09/31/11. The add-marriage-certificate-date expert reports back that this date is earlier than the birthdate recorded for one of the individual's. Upon examination, Betsy realizes she meant to type 09/31/21. This time the data is validated. The marriage date expert reports back to Betsy that a marriage date from a Bible source already is associated with the primary individuals. It differs from this marriage certificate date. The marriage certificate date will be given precedence. The database is instructed to add the data fields. Betsy receives an update complete message. An ownership record recording Betsy, is created for each of the added data items.

Two days later, Joan, an associate of Betsy signs on to the genealogical system. She is involved her own independent research. Joan is worried that she entered

the wrong data three days ago. She requests the database menu, selecting the query option. She displays all of the source record data recorded for a particular individual. It just so happens to be one of the primary individual's to which Betsy added information. Joan doesn't remember typing in the marriage certificate information. She assumes this was one of her mistakes. Entering the delete database function, she attempts to delete the marriage certificate source information. The experts for each of the fields return security violation messages, informing Joan that she is not the owner of this data.

4.6. Example of an Expert

This section presents a high level, skeletal look at a typical expert. The principles illustrated in this example apply to the other experts as well. The example expert is presented in a 'pseudo' language. Interfaces with the user and the database sub-system are indicated.

Add-Birth-Certificate-Birthdate Expert

Start:

- 1) Examine the birth certificate birth date field.

If field is blank then:

- a) tell the user front-end to highlight

the field,

- b) send error message to user,
- c) request user to re-enter the field,
and
- d) go back to start.

2) Execute Validate-Date Expert

If date is invalid then perform steps 1a-1d.

3) Execute Verify-Birthdate Expert

If date is not semantically correct then:

- a) perform steps 1a-1b,
- b) display the dates in conflict,
- c) request user to verify the date in
relation to the dates in conflict,
- d) request user to re-enter the date,
along with associated text describing
the inconsistencies, or cancel this
add,
- e) if date entered, go back to start.

4) Ask the database sub-system if there are any
data values associated with this birthdate
attribute.

- a) NO - Tell database sub-system to add
the birthdate, marking it as the
precedent birthdate. Go to EXIT

b) YES - Request the database sub-system to provide all tuples containing the associated birthdates.

5) Execute Birthdate-Precedence Expert

Database sub-system is instructed to add the new data item. The birthdate that was specified as having precedence is marked in the database.

EXIT:

Validate-Birthdate Expert

The input to this expert is a date. This expert verifies that the date is in proper form. Genealogical dates include many forms, such as Gregorian and Julian. Any date that does not fall into one of the prescribed forms is flagged as being in error.

Verify-Birthdate Expert

The input to this expert is a date. This expert verifies that the date is semantically correct. The date is compared with the existing dates associated with the target primary individual. Logic determines whether a date is inconsistent. For example, a birth date must be

earlier than all other dates (except possibly the death date).

Birthdate-Precedence Expert

The input for this expert is two or more dates. This expert determines which of the dates has highest precedence. The rules used are listed in the order of precedence, 1) being the highest.

1. The date of birth from a birth certificate has precedence over birth date from a death certificate, tombstone, or Bible.
2. The date of birth from death certificate has precedence over birth date from tombstone or Bible.
3. The date of birth from Bible has precedence over birth date from tombstone if the Bible entry was made at the time of the birth.
4. The date of birth from tombstone has precedence over birth date from unsupported sources, such as word of mouth.

4.7 Summary

Professional genealogists demonstrate a need for a tool that assists them with their decision processes and distributes hard to obtain knowledge. The use of expert

system technology in genealogy is promising. Difficulties in applying this technology were encountered because of the complexity of the genealogical knowledge and data. Chapter 5 presents a detailed discussion of the difficulties. Many of the originally designed features were discovered to be beyond the scope of this thesis.

This thesis proposes the concept of experts. Experts are modular, independent decision making units. The experts covered in this design make decisions on the validity, integrity, and precedence of data items. Each data item has an associated add, delete, and update expert. The experts act as a function on the datum, returning decisions and knowledge to the database and user front-end systems. By separating the genealogist's knowledge into independent modules, many of the original design problems are addressed.

CHAPTER 5 - DESIGN PROBLEMS ENCOUNTERED

5.1 Introduction

This chapter documents the early design efforts in integrating expert system technology into a genealogical information system. The goal of the initial design efforts was to provide professional genealogists with an intelligent tool that would assist them with their research and publishing endeavors. After the design process began, it was determined that current, commonly used expert system approaches did not address the difficulties that arose. This chapter describes the problems that led to the change in the design scope. Section 5.2 describes the original conception of the expert system. Section 5.3 discusses the problems that were encountered. This is followed by a summary in section 5.4.

5.2 Original Expert System Design

An understanding of the original expert system design is necessary before presenting the problems that arose. The original expert system was viewed as a single system which contained an expert system driver, an inference engine, and an explanation generator. The expert system driver interfaced with the user front-end and the database

system. The inference engine was the main component of the expert system. It contained a controller, a scheduler, an interpreter, an intermediate storage facility, and a knowledge base. An overview of this original system is depicted in Figure 5.1.

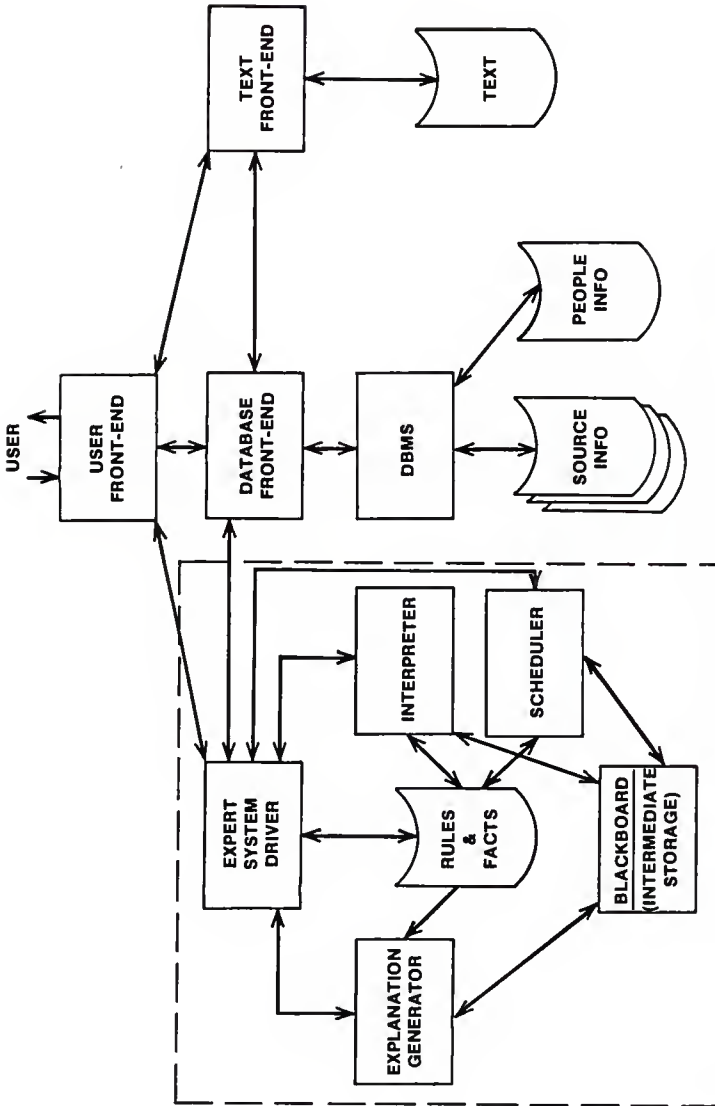


FIGURE 5.1 ORIGINAL SYSTEM DESIGN

The inference engine was to have provided a high powered, rule processing system. It needed the capability to set goals, hone in on the rules needed to meet those goals, and derive solutions. This would have required the incorporation of search techniques most suited to the genealogical data and knowledge representation. Possible options included: state-space, blind, or heuristic searches, generate and test methods, or solution space abstraction [BU86].

The expert system controller was to have incorporated one of these search strategies. It directed the flow of data and solutions by setting strategies and goals. The scheduler's purpose was to keep track of the solution agenda. It determined what rule or action should be executed next. The interpreter executed the rule or action chosen by the scheduler. This unit was responsible for binding the data from the genealogical database and knowledge base to the chosen rule. It was also responsible for validating the relevance conditions of rules. The knowledge base was composed of the rules and facts supplied by the contributing genealogist. It was envisioned to be a mass of rules organized into logically related packets of rules. The explanation generator provided detailed explanations of each derived decision.

The intermediate storage acted like a 'blackboard,' containing the rules and agenda chosen by the scheduler, the partial solutions and the rules used to derive these solutions, and the general strategies established.

The system communicated with the database system through the expert system driver. It was also, responsible for making the requests for the database tuples.

5.3 Problems Encountered During Design Phase

As the design process proceeded, it became clear that the design would have to be altered due to personal computer limitations and the problems inherent to genealogical knowledge. Section 5.3.1 describes the problems due to the complexities and size of the genealogical knowledge base. Section 5.3.2 discusses the problems due to the personal computer requirement.

5.3.1 Large and Complex Knowledge Base

A professional genealogist uses rules-of-thumb, strategies, local history and laws, and general genealogical knowledge to determine identities and relationships from the diverse source documents collected. To mechanize this process, the expert system has to incorporate this "expert" knowledge in the form of rules. For each field of information in the source document,

there are a set of rules governing the implications which can be derived from this field, as well as from the comparisons of this field with other data in the genealogical database.

The expert system must continually analyze and re-analyze the information entered from these source documents. A source document that initially provides no clues, might provide a wealth of information when used in conjunction with another source document. The analysis process is iterative, repeating itself for each new source document entered into the system. Each field of a source document must be compared not only with each other, but also with the data associated with other source documents. The possible combinations of data to be compared are extremely large. The information derived from this iterative process provides inferences and clues to the identity of individuals and the determination of relationships.

The knowledge must be structured in such a way to cover the possible comparisons needed without going down multiple trails. In the original design process, the volume of the inter-related genealogical rules-of-thumb and knowledge provided many possible decision routes (using either forward or backward chaining). This made the

decision process inefficient. It was difficult to structure the rules provide efficient searching.

The original design proposed packets of rules with logical break points established, allowing the user to modify the decision process. The complexities and inter-relationships of the rules made it difficult to structure the rules into modules of related knowledge. The same rule may be needed in a multitude of decisions. The problems caused by this loss of modularity are:

1. The intricate inter-relationships of the rules resulted in the rules losing not only modularity, but also lucidness. If a rule can interact with many other rules in unstructured ways, it becomes difficult to determine how the rule affects the overall behavior of the system [MA86].
2. The loss of modularity made it extremely difficult to design a way for rules to be added or deleted from the knowledge base. The rule base's integrity could possibly be impacted by the addition, modification, or deletion of rules.
3. The loss of modularity made it difficult to devise a method to allow the user to temporarily override the

decision process. Logical break points could not be established.

5.3.2 Personal Computer Requirement

The original design dictated that the system must be designed to run on an affordable, personal computer. The personal computer requirement provided several limitations:

1. Personal computers have a limited amount of storage. It will require a tremendous amount of memory to store the mass of genealogical rules, as well as the source data that a professional genealogist collects. A typical, affordable, hard-disk personal computer system will not hold this data.
2. The expert system tools available for personal computers do not have a high level of sophistication. Each tool utilizes a particular search strategy. Thus, the choice of a search strategy is limited to the availability of tools. In addition, the lack of a "skeletal" expert system tool meant that the design effort would also have to include functions that a tool could have provided.

The intent of the original design was to utilize an existing tool. The span of the design period did not

allow for the design of a tool or base system in addition to the other requirements of the genealogical information system. For these reasons some of the features mentioned above will be removed from the scope of this design.

5.4 Summary

As a result of the original design efforts, it was determined that current, commonly used expert system techniques were not applicable for a large, complex, inter-related rule base. In order to modularize the knowledge, another look was taken at the genealogical process. This process can be broken down into four main categories:

- 1) decisions on data,
- 2) identifying individuals,
- 3) determining relationships, and
- 4) developing future strategies.

It was determined that a modular approach to the system might address some of the inherent problems. From this, the concept of experts arose. The expert system would consist of a series of experts. The problem areas would be broken down into a hierarchy of these experts, with the most specific experts at the bottom. To analyze such a concept, this design addressed the simplest category of

genealogical decision process, the decisions on data. For a complete description, refer to Chapter 4.

CHAPTER 6 - IMPLEMENTATION

6.1 Introduction

This chapter addresses the implementation of a prototype of a genealogical data base system. This prototype was modeled after the design of the data base sub-system of the Professional Information Genealogical System. The data base design was addressed in detail in another Kansas State University thesis [BE88]. The prototype was implemented as a stand-alone system. The interfaces to an expert system and a text management system were not developed. The primary purpose of the prototype was to implement and illustrate the data base design that is the basis of the Professional Genealogical Information System's Data Base Sub-system.

A brief description of the data base design is given in Section 6.2. A brief overview of the prototype is presented in Section 6.3. The prototype functions are discussed in Section 6.4, followed by a summary in Section 6.5.

6.2 Data Base Design

The data base design is based on the concept of source documents. The premise is that all information collected by a genealogist comes from a source document of one type

or another. The source documents collected by the genealogist provides information needed to determine identities and relationships. Among the formal source documents are birth, marriage, and death certificates. Informal sources would include Bibles, tombstones, etc..

A data base file was created for each of the major categories of information that can be gathered from the source documents. For the prototype, the categories include birth, marriage, and death information. Data base files were established to hold the names of individuals, the relationships of the individuals, and the source document descriptions. As part of the original requirement of the Genealogical Information System, a unique source document number and person number are provided by the system. Two data base files were created to maintain the availability of the numbers to be assigned. The data base structure for each of the above described data base files is in presented in Table 6.1. The normalization process determined the final, complex data base design.

A single source document can provide multiple categories of information, as well as information about multiple people. For example, a death certificate can provide

birth, marriage, and death information about the deceased, and the identities of the deceased spouse and parents.

All birth, marriage, and death certificates entered into the prototype system have an established format. The data base file structures have been established for defined fields. Additional fields may not be added to the source document formats without changing the data base file structures. In reality, these source documents could vary from one locality to another. Refer to Chapter 7 for a discussion on possible enhancements of the system.

Linkages had to be established between the data base files to retain the relationship of the data provided by the source documents. These linkages were implemented via the use of key values. The data base management package selected was dBASE III Plus. It provided the capabilities of multiple attributes as a key expression for an index file, of multiple index files active at one time, and of relationships between multiple data bases. The prototype utilizes these capabilities. Unique source and person numbers were set up as key values of the data base files. The prototype establishes four data base work areas with appropriate index file(s) active in each. dBASE III Plus relationships are established between the indexed, data base work areas.

The prototype does not address null values in the key fields (values must be entered into the key fields). The indices for each data base file are listed in Table 6.2.

Table 6.1 Data Base Structures

Database	Field Name	Type	Width
PERSON	PERS_NO	Character	6
	PR_F_NAME	Character	15
	PR_M_NAME	Character	15
	PR_L_NAME	Character	20
SOURCE	SRC_NO	Character	6
	PERS_NO	Character	6
	SRC_TYPE	Character	2
	DATE_SR_FD	Date	8
	CITY_SR_FD	Character	20
	CNTY_SR_FD	Character	20
	ST_SR_FD	Character	2
	INT_FD	Character	3
	DATE_SR_ET	Date	8
RELATION	SRC_NO	Character	6
	PERS_NO	Character	6
	REL_PRS_NO	Character	6
	RELATION	Character	10
BIRTH_IN	SRC_NO	Character	6
	PERS_NO	Character	6
	BIRTH_DATE	Date	8
	BIRTH_CITY	Character	20
	BIRTH_CNTY	Character	20
	BIRTH_ST	Character	2
	BIRTH_FAC	Character	10
	BIRTH_RACE	Character	10
BIRTH_SEX	Character	1	
DEATH_IN	SRC_NO	Character	6
	PERS_NO	Character	6
	DEATH_DATE	Date	8
	DEATH_CITY	Character	20
	DEATH_CNTY	Character	20
	DEATH_ST	Character	2
	DEATH_CAUS	Character	15
	DEATH_RACE	Character	10
	DEATH_SEX	Character	1
	BURY_DATE	Character	8
	BURY_CITY	Character	20
	BURY_CNTY	Character	20
	BURY_ST	Character	2
BURY_FAC	Character	15	

Table 6.1 - Data Base Structures Continued

Database	Field Name	Type	Width
MARRIAGE	SRC_NO	Character	6
	PERS_NO	Character	6
	MARRY_DATE	Date	8
	MARRY_CITY	Character	20
	MARRY_CNTY	Character	20
	MARRY_ST	Character	2
	MARRY_FAC	Character	10
	MARRY_RACE	Character	10
	MARRY_SEX	Character	1
MARRY_AGE	Character	3	
SRCNO	KEY	Character	7
	NO	Character	6
PRSNO	KEY	Character	7
	NO	Character	6

Table 6.2 - Data Base Index Structures

Database	Index Name	Data Base Fields
BIRTH_IN	BSRC	SRC_NO
	SRCPERS	SRC_NO+PERS_NO
SOURCE	PERSTYPE	PERS_NO+SRC_TYPE
	SRC	SRC_NO
	SPERS	PERS_NO
MARRIAGE	MSRC	SRC_NO
	MSRCPERS	SRC_NO+PERS_NO
PERSON	NAME	PR_L_NAME+PR_F_NAME+PR_M_NAME
	NAME2	PR_L_NAME+PR_F_NAME
	NAME3	PR_L_NAME+PR_M_NAME
	PERS	PERS_NO
DEATH_IN	DSRC	SRC_NO
	DSRCPERS	SRC_NO+PERS_NO
RELATION	SRC	SRC_NO

6.3 Prototype Overview

One of the requirements of the Professional Information Genealogical System was that the system must run on a personal computer. The personal computer used for the prototype was an AT&T 6300 (IBM-compatible). Various data base management systems that are compatible with the AT&T 6300 were investigated. dBASE III Plus was chosen as the data base management system for the prototype because of a number of reasons: it appeared to provide the functionality needed for the prototype, it was one of the more sophisticated data base management systems investigated, and the software was available to the developers. The dBASE III Plus code for this implementation is included in Appendix A.

The system automatically generates a unique number for each person added to the genealogical data base. A unique number is also automatically generated for each source document entered. As source documents are deleted from the system, the numbers are freed up and available to be used again. The unique, system generated numbers allow information from source documents to remain logically related. This linkage gives the genealogist the flexibility to add or delete information about individuals

without having to renumber or reorganize the data already collected.

The precedence of the genealogical data is implemented via program intelligence or through online interaction with the user. A person can have multiple source records which provide the same category of information. Precedence of these source records is achieved by setting the source type field in the SOURCE data base file to a code that indicates priority. Information for the primary individual on the source document would be linked to a source entry marked with the source type 'xC' (x = M for marriage certificates, B for birth certificates, and D for death certificates). Information for the secondary individuals on the source document would be linked to a source entry marked with the source type 'xR' (M, B, or D). For example, the birth information derived from a person's birth certificate would be marked with a source type of 'BC', whereas a person's birth information that is derived from his/her child's birth certificate would be marked with a source type of 'BR'. Similarly, for marriage information, 'MC' has precedence over 'MR', and for death information, 'DC' has precedence over 'DR'. The system at this time sets no precedence for multiple birth, marriage, or death certificates entered.

6.4 Prototype Functions

The original design proposed the establishment of a user front end sub-system. This sub-system was not implemented. A user interface was integrated into the system. The user interface is accomplished through online interaction and through the use of menus. The prototype is composed of a hierarchy of functional menus. The primary menu functions include: help information, add source records, delete source records, update source records, and query/reports.

6.4.1 Help Information Function

The Help Information Function provides tutorial information on the Professional Information Genealogical System. Descriptions of each of the system functions are given. This information includes requirements and restrictions for each source type.

6.4.2 Add Source Record Function

The Add Source Record Function allows the user to add source document information to the genealogical data base files. The prototype prompts the user to select the source document type to be entered. The appropriate source document entry screen is presented. After the user

has typed information into the fields on the screen, the system validates the data.

A source record will not be accepted unless the initials of the person entering the source document are given. This information will be used later for data integrity controls.

All date fields are checked to ensure they contain valid dates. The primary date of each source document type must be entered for the source document to be accepted. For example, the birth date must be entered for a birth certificate to be accepted.

For each individual listed on the source document, the last name must be given in combination with the first, and/or middle name. The first or middle name can be an initial.

After the source data has passed verification, the information is added to the appropriate genealogical data base files. For illustration, an add birth certificate scenario is given.

For the primary person (person born):

The user is asked, "Is this a person already entered in the system." If the answer is no or unsure, then the

system searches the person data base for all names which match the name combination given. This is facilitated through multiple indices established to handle the various first, middle and last name combinations. The matches may be viewed online or printed to a printer. The user must choose either a matched person (by specifying a person number and a source number) or state that the person is indeed new.

A unique source number is generated. If the person is new, a unique person number is created. An entry is added to the SOURCE data base file, relating this source document to the individual. The source type is set to 'BC', indicating that this birth source entry came from a birth certificate and not from other types of documents. The unique source number is used to link the source document to all of the other data base file entries created due to information from this source document.

If the person is new, an entry for this person is added to the PERSON data base file.

An entry is added to the BIRTH_IN data base file, providing birth information for the person. This entry is linked via source number to the originating source document and linked via person number to the person's name.

For the secondary people (parents):

The process is very similar for the secondary people listed on the birth certificate, with minor differences. If parents are listed on the birth certificate, it must be determined whether they already exist in the data base or are new. If new, unique source numbers are created and entries are added to the PERSON data base file. Entries are added to the SOURCE data base file, relating these secondary people to this birth certificate document. If the parents had birth dates listed on the birth certificate, the source type is set to 'BR', indicating the birth information came from a source other than the person's own birth certificate. An entry is added to the RELATION data base, specifying the relationship of this secondary person to the source document's primary person (the primary person has a relationship of 'SELF' to himself).

6.4.3 Delete Source Record Function

The Delete Source Record Function allows the user to delete source records from the genealogical data base files. The prototype prompts the user to select the type of source document to be deleted. The source document's

primary person's name is requested. The name combination requirements are the same as for the add function.

The PERSON data base file is searched for all names which match the name combination given. This is facilitated through multiple indices established to handle the various name combinations. The matches may be viewed on-line or sent to a printer. The user must choose a matched person. This is done by selecting a person number and associated source number from the list presented.

Next, the system verifies whether the primary person of this source document to be deleted is linked to any other source documents. If the person is not associated with other source documents, the person is deleted from the PERSON data base file.

The source record entry for this person is deleted from the SOURCE data base file. Any associated information from this source document that was added to the various data base files, is deleted. For example, in the case of a birth certificate, the birth information record is deleted from the BIRTH_IN data base file. The relation of this person to the source document's primary person (in this case, the same) is deleted from the RELATION data base file.

The system repeats this process for the secondary individuals associated with the source document. It is important to note, that the system will only delete an individual from the PERSON data base file if the individual is not associated with any other source records.

The result of the Delete function is the deletion of all information that was added to the system via the add function.

6.4.4 Update Source Record Function

The Update Source Record Function allows the user to modify a source record that was previously added to the system. The prototype prompts the user to select the source document type to be updated. The source document's primary person's name is requested. The name combination requirements are the same as for the delete function. The search process is the same as for the delete process.

The user is presented with all of the entries with names that match the name combination given. The user selects the person number and source number of the primary person who matches. The system uses this person and source number to reassemble the source document information in the original entry screen format.

The RELATION data base file is used to obtain the information on the source document's secondary people. The following is the process to reassemble the information for the secondary people listed on a birth certificate. The source number and the infant's person number is used to link to the RELATION data base. For each secondary person, there will be an entry with the secondary person's person number and their relationship to the primary person. If the parent's had birth information listed on their child's birth certificate, they will have birth information in the BIRTH_IN data base file. Their secondary person numbers are used in conjunction with the source number to link, in this case, to BIRTH_IN data base file.

After the source document is reassembled, the user is allowed to change any field on the screen. All of the associated fields of information are updated in the appropriate data base files. No linkages are changed as the result of the update function.

6.4.4 Query/Report Function

The Query/Report Function allows the user to do queries and produce reports. The prototype prompts the user to select the query sub-function or the report sub-function.

The query sub-function allows the user to view any source record that has been added to the system. The system prompts the user to select the type of source document to be viewed. The source document's primary person's name is requested. Following the same procedure as described in the functions mentioned above, all names which match the name combination are given. The user again chooses a person number and the associated source number. This information is used to create linkages between the data base files and reassemble the source document.

The report sub-function provides two type of reports: a report of ancestors and a report of descendants. The user is prompted to select a report type and to enter the name of the person for whom the report will be based. The system, via interaction with the user, determines the individual's person number.

Ancestor Report

The ancestor report produces an individual's ancestry. The person's name is printed at the top, followed by the spouse's name (if there is one). The spouse is determined by linking information in the RELATION, SOURCE, MARRIAGE and PERSON data base file. No ancestral search will be performed for the spouse.

The SOURCE data base file is searched for a 'BC' source entry, which indicates a birth certificate for the individual. If multiple 'BC' entries for this individual exist, the search will stop after the first one is found. If a 'BC' entry is not found, 'MC', then 'DC' source entries for this individual are searched. Once a source record has been found, the SOURCE data base entry is linked to the RELATION data base via the person number and the source number. Through this linkage, the person numbers of the parents can be obtained. The name of each parent obtained is printed. The search process for their children begins. The process is a reiteration of the steps outlined above. The ancestor report is printed in the format of a tree.

Descendant Report

The Descendant Report produces a descendant tree. The person's name is printed at the top, followed by the spouse's name (if there is one). The spouse is determined by linking information in the RELATION, SOURCE, MARRIAGE and PERSON data base file. No descendant search will be performed for the spouse.

The SOURCE data base file is searched for each 'BR' entry, using the individual's person number. There will be a 'BR' entry for each birth certificate for which this

individual is listed as a parent. The individual's person number and the source numbers are used to link to the relation person number and source number fields of the RELATION data base file. This linkage provides the person numbers of the individual's children. For each child found, the name is printed, along with their spouse. The search process described in this paragraph is repeated for each child found, producing a tree-like report.

6.5 Summary

A prototype of the data base sub-system of the Genealogical Information System was implemented. It is based on a data base design addressed in another Kansas State University thesis [BE88]. The data base design is centered around the concept of source documents. A data base file was created for each of the major categories of information that can be gathered from source documents. Data is added, deleted, updated and viewed in the form of source records. To facilitate this, the system generates unique source and person numbers. These numbers are the key fields used to relate the data in the data base files. The index schemes provide for effective and efficient searches.

The prototype was developed using the dBASE III Plus data base management system. A discussion of possible

enhancements to this implementation is presented in Chapter 7. The dBASE III Plus source code for this implementation is located in Appendix A.

CHAPTER 7 - CONCLUSIONS AND EXTENSIONS

7.1 Introduction

This chapter presents the conclusions of this research effort and future extensions. Section 6.2 describes the conclusions derived from the requirements and design of the Experts and from the implementation of a prototype of the Professional Information Genealogical System data base design. Section 6.3 presents future extensions which would extend the concept presented here.

7.2 Conclusions

7.2.1 Expert System Design

The scope of this thesis was to address the concept of employing expert system technology, in the form of experts, in a database information system. The experts were to provide intelligence to the database sub-system. The experts would provide these categories of information: decisions on a datum, identification of identities, determination of relationships, and derivation of solutions. This paper only addressed the design of experts which act as functions on a datum.

From the original and from the resulting design efforts, several conclusions were reached concerning expert systems

and the problem domain of genealogy. The problems that occurred during the original design effort will not be discussed in this chapter. For a detailed discussion, refer to Chapter 5.

Genealogy is not an ideal problem domain for the development of an expert system, as was previously thought. Research and studies have shown that successful, practical expert systems so far have each addressed a specialized task. Genealogy is a complex collection of tasks. Genealogical experts utilize not only genealogical knowledge and rules of thumb, but also local history, customs and laws, common sense knowledge about the world and a good deal of intuition. They often have to deal with judgmental, inexact information. It is extremely difficult to represent intuitive and inexact knowledge.

Although genealogy may not be a practical environment for expert system development, much was learned from the design attempt. Complex, inter-related knowledge bases should be restructured into modular units of knowledge. These units can then be built into hierarchies of functions, yet still retain the benefits of modularity.

The original requirement of a personal computer is not practical. Professional genealogists utilize a vast amount of knowledge to make their decisions. They also

collect extremely large amounts of data in the form of source records. The amount of space required to hold the software, the knowledge base and the genealogical data base is more than a typical, "affordable" personal computer would have. A solution would be to implement the expert system on a medium size computer and allow multiple professional genealogists to link their "affordable" personal computers into it via telecommunications.

The expert system skeletal tools available for personal computers are very simplistic and have non-sophisticated search strategies. Most of the systems have restrictions on the number of rules that may be stored. Three systems that I reviewed limited the number of rules to 800 or less. The genealogical knowledge base, once complete, would be larger than this. The packages appear to be more suited as learning tools. There are expert system tools being used on larger computers, though, which provide the functionality and versatility that a design of this type would need.

7.2.2 Prototype Implementation

From the implementation of a prototype of the Professional Information Genealogical System data base design, it was determined that a data base management system other than

dBASE III Plus should be utilized. The following restrictions of dBASE III Plus proved cumbersome for the implementation: only seven index files can be active at one time, and only one relation can be established between any one active data base file to another. The data base design was based on multiple active relationships across up to four data bases at a time. The capability to position data base pointers in one data base and save the position while linking to another data base was needed. dBASE III Plus would not allow this.

Another requirement of the design was the establishment of multiple indices to provide for efficient searches. In some instances, multiple indices needed to be active at the same time. For example, in adding a birth certificate, nine indices ideally need to be active. dBASE III Plus only allows seven to be active at any one time.

7.3 Extensions

7.3.1 Expert System Design

This section discusses possible extensions to the requirements and design of this thesis. Due to the problems that arose during the original design process, the resulting design focused on only one component that

would be needed for the genealogical expert system, data experts. Additional experts need to be designed that ideally would provide search strategies, decisions on relationships and identities, derivation of research strategies, and determination of precedence.

Efficient search strategies need to be investigated. The genealogical knowledge base presents unique problems. The inter-relationship of the rules provides for many possible blind trails. Without an effective search strategy, the decisions returned by the system could be incorrect and the response time unacceptable.

Selection of an expert system skeletal tool must be made. This tool should incorporate the search strategies selected in the previous paragraph. The tool should provide the following functionalities:

- an inference engine, for reasoning with the facts and rules,
- an explanation generator,
- a scheduler, and
- an interpreter.

A user front-end needs to be developed. Ideally, this front-end should incorporate natural language processing. Possibilities would be the establishment of a syntax and

grammar. Users should be given the capability to utilize screen menus, or interface with the system in English-like sentences (ad hoc queries). This front-end should interface with the expert sub-system, the data base sub-system, and the text sub-system.

A scheme should be established to provide certainty or probability factors to each decision made. This would be useful in dealing with inexact information. Additionally, intelligence needs to be developed to handle null values in key fields in the genealogical data base files.

A knowledge acquisition system would be extremely helpful. In building an expert system, the largest amount of time is spent on acquiring and representing knowledge. This is a new area of research in expert system development.

A data base management system needs to be selected that has the capability to interface with an expert system. The DBMS should include a supporting language such as PROLOG, SQL, or LISP, to facilitate ad hoc capabilities and flexibilities.

A soundex capability should be developed. Soundex is an indexing system used by the National Archives for some censuses, in which use of a numerical code facilitates the identification of names appearing under variant spellings.

7.2.2 Prototype Implementation

The prototype only included the basic functionality that would be needed for the data base portion of the Professional Information Genealogical System. The following extensions need to be implemented:

- Process more source document types,
- Better interaction with the user,
- Handle informal source information types such as family Bible, tombstone, etc.,
- Provide intelligence to determine precedence,
- Capability of ad hoc requests, and
- Extended report capabilities.

7.4 Summary

This research endeavor supported the premise that practical expert systems are not applicable to all problem domains. The successful expert systems developed to date, all address a specialized task. Genealogy is a complex collection of tasks. Intuition and inexact knowledge is heavily relied upon. For these reasons, it is not an ideal problem domain for an expert system to be developed.

Much can be learned about a problem domain, such as, genealogy. To establish a successful expert system for

genealogy, a hierarchy of 'experts' should be implemented, breaking genealogy down into specialized tasks. For expert systems to be widely and successfully implemented in the business arena, this same approach will be necessary.

REFERENCES

- [AL80] Albright, L. and Leary, H. M. "Research Strategies." North Carolina Research Genealogy and Local History." eds., H. F. M. Leary, and M. R. Stirewalt, Saline, MI: McNaughton and Gunn, 1980, pp.1-39.
- [BA85] Basu, Dipankar. "The Quest for the Intelligent Data Base." Business Software Review. (October 1985), pp.62-63.
- [BE88] Bailey, Ellen J. "Design of a Professional Genealogical Information System: Including Navigation from an Unstructured Database to a Structured Database." Master's Thesis, Kansas State University, 1987.
- [BI86] Bic, L. and Gilbert, J. P. "Learning from AI: New Trends in Database Technology." Computer (March 1986), pp.44-54.
- [BR83] Brachman, Ronald J. "What are Expert Systems?" in Building Expert Systems. eds., F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, Reading, MA: Addison-Wesley, 1983, pp.31-58.
- [BU83] Buchanan, Bruce G. et al. "Constructing an Expert System." in Building Expert Systems. eds., F. Hayes-Roth, D. A. Waterman, D. B. Lenat, Reading, MA: Addison-Wesley, 1983, pp.127-168.
- [DU81] Duda, Richard O. and Gaschnig, John G. "Knowledge-Based Expert Systems Come of Age." Byte, (September 1981), pp.238-279.
- [HA85] Hartzband, David J. and Maryanski, Fred J. "Enhancing Knowledge Representation in Engineering Databases." Computer (September 1985), pp.39-46.
- [HO71] Holland, Derek. Genealogical Research Standards. Salt Lake City, UT: Westman Publishing, Inc., 1971.
- [HR83] Hayes-Roth, Frederic, Waterman, Donald A., and Lenat, Douglas B. "An Overview of Expert Systems." in Building Expert Systems. eds., F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, Reading, MA: Addison-Wesley, 1983, pp.3-30.

- [KR83] Kroenke, D. Database Processing. Chicago, I.L.: Science Research Associates, Inc, 1983.
- [KS85] "Genealogical Software." Kansas Library Automation News. number 7 (March 1985), pp.6-13.
- [LE86] Leary, F. H. M. Personal interview 19, May, 1986.
- [MA86] Mark, William "Rule-Based Inference in Large Knowledge Bases." USC/Information Sciences Institute, 1986.
- [MI86] Minter, Joan R. Personal interview 13, April, 1986.
- [SK84] Skidmore, William F. "Genealogy and the Home Computer." Magazine of Virginia Genealogy, vol. 22, no.3 (August 1984), p.43.
- [ST83] Stefik, Mark et al. "The Architecture of Expert Systems." in Building Expert Systems. eds., F. Hayes-Roth, D. A. Waterman, D. B. Lenat, Reading, MA: Addison-Wesley, 1983, pp.89-126.
- [WA86] Waterman, D. A. A Guide to Expert Systems. Reading, MA: Addison-Wesley Publishing Company, Inc., 1986
- [WI84] Wiederhold, G. "Knowledge and Database Management." IEEE Software (January 1984), pp.63 - 73.
- [WK86] Walker, Adrian. "Knowledge Systems: Principles and Practice." IBM Journal Research Development, vol.30, no.1 (January 1986), pp.2-13.
- [WL85] Wilson, Maria. "A Study of Null Values." Master's Thesis, Kansas State University, 1985.

APPENDIX A - dBASE III PLUS CODE

```

*****
*
*                               P I G . P R G
*-----*
*   TYPE:  program library
*
*   CALLED FROM:  This is the starting program
*-----*
*
*   PROGRAMS INVOKED:  INITVAR
*                     INITSRV
*                     INITSC
*                     MMENU
*-----*
*
*   LOGIC:  This is the initial program executed when
*           entering the PIG system.  Global variables
*           are initialized and control is passed over
*           to the main driver, MMENU.prg
*****
*
SET STATUS OFF
set bell off
SET SCOREBOARD OFF
RELEASE ALL
SET ECHO OFF
SET TALK OFF
@ 4,4 TO 18,75 DOUBLE
@ 6,8 TO 16,71 DOUBLE
@ 10,18 SAY "PROFESSIONAL INFORMATION GENEALOGICAL"
@ 12,32 SAY "SYSTEM"
CLEAR GETS
store " " to CONTVAR
@ 23,10 SAY "Type 'C' to continue . . ." get CONTVAR
read
do while CONTVAR <> "C"
  .store " " to CONTVAR
  .clear gets
  @23,10 say "Type 'C' to continue ..." get CONTVAR
  read
enddo
*
*   set variables & give control to main menu program
*
DO initvar
DO initsrv
DO initc
DO mmenu.prg

```



```

*****
*
*           M M E N U . P R G
*-----*
* TYPE:   program library
*
* CALLED BY:  PIG
*
* PROGRAM INVOKED:  HELPINFO
*                   ENTERSRC
*                   DELTSRC
*                   RPTS
*                   UPDATREC
*                   ADHOC
*
* GLOBAL VARIABLES: none
*
* LOGIC:  THIS IS THE MAIN DRIVER PROGRAM.  THE MAIN
*         MENU IS DISPLAYED.  DEPENDING ON THE FUNCTION *
*         CHOSEN, THE CORRESPONDING PROCEDURE IS CALLED.*
*****
sat talk off
SET ECHO OFF
STORE SPACE(1) TO OPT
DO WHILE OPT <> "6"
*
*   draw master manu on screen
*
  claar
  sat bell off
  STORE SPACE(1) TO OPT
  @ 2,19 to 6,60
  @ 7,19 to 20,60
  @ 4,21 say "PROFESSIONAL INFORMATION GENEALOGICAL"
  @ 5,37 say "SYSTEM"
  @ 9,30 say "*** Main Menu ***"
  @ 12,28 say "(1) Halp Information"
  @ 13,28 say "(2) Add Sourca Racords"
  @ 14,28 say "(3) Deleta Sourca Racords"
  @ 15,28 say "(4) Updata Sourca Racords"
  @ 16,28 say "(5) Quarry/Raports  "
  @ 17,28 say "(6) Exit      "
  clear gets
  @ 23,28 say "Entar Selaction  " gat OPT
  read
  do while opt < "1" .or. OPT > "6"
    store " " to OPT
    clear gets
    @ 23,28 say "Entar Selaction  " gat OPT
    read
  enddo
*
* CALL APPROPRIATE PROCEDURE BASED ON OPTION SELECTED
*
do case
  case OPT = "1"
***    Help Information Procadura          ***
    do halpinfo
  case OPT = "2"

```

```
***      Enter Source Record Procedure      ***
do entersrc
case OPT = "3"
***      Delete Source Record Procedure      ***
do deltsrc
case OPT = "4"
***      Update Records                      ***
do updatrec
case OPT = "5"
***      Generate Reports                    ***
do rpts
case OPT = "6"
***      Exit from PIG system                ***
exit
endcase
enddo
```

 * I N I T S C R V . P R G *

* TYPE: Program Library *
 * * * * *

* CALLED FROM: FIG, bthcert, mrgcert, dthcert *
 * * * * *

* GLOBAL VARIABLES: dte, city, cnty, st, fec, sex, *
 * rece, fneme, mname, lname, *
 * ddte, dcity, dcnty, dst, *
 * ceue, bdate, bcity, bcnty, bst, *
 * bfac, mdte, ege, bage, bsex, *
 * brace, bfname, bmname, blname, *
 * ffname, fname, flname, mfname, *
 * mname, mlname, fbname, fbname, *
 * fblname, mbfname, mbname, *
 * mblname *
 * * * * *

* LOGIC: This routine initializes and establishes *
 * the type, of the variables associated with *
 * BIRTH_IN, PERSON, DEATH_IN, and MARRIAGE *
 * date base files. *
 * * * * *

 use BIRTH_IN.dbf
 append BLANK
 store BIRTH_DATE to dte
 store BIRTH_CITY to city
 store BIRTH_CNTY to cnty
 store BIRTH_ST to st
 store BIRTH_FAC to fec
 store BIRTH_SEX to sex
 store BIRTH_RACE to rece

use PERSON.dbf
 append BLANK
 store PR_F_NAME to fname
 store PR_M_NAME to mname
 store PR_L_NAME to lname
 store PR_F_NAME to bfname
 store PR_M_NAME to bmname
 store PR_L_NAME to blname
 store PR_F_NAME to ffname
 store PR_M_NAME to ffname
 store PR_L_NAME to flname
 store PR_F_NAME to mfname
 store PR_M_NAME to mname
 store PR_L_NAME to mlname
 store PR_F_NAME to fbname
 store PR_M_NAME to fbname
 store PR_L_NAME to fblname
 store PR_F_NAME to mbfname
 store PR_M_NAME to mbname
 store PR_L_NAME to mblname

use DEATH_IN.dbf
 append BLANK
 store DEATH_DATE to ddte

```
store DEATH_CITY to dcity
store DEATH_CNTY to dcnty
store DEATH_ST to dst
store DEATH_CAUS to caus
store BURY_DATE to bdate
store BURY_CITY to bcity
store BURY_CNTY to bcnty
store BURY_ST to bst
store BURY_FAC to bfac
*****
use MARRIAGE.dbf
append BLANK
store MARRY_DATE to mdte
store MARRY_RACE to brace
store MARRY_AGE to age
store MARRY_AGE to bage
store MARRY_SEX to bsex
store MARRY_RACE to brace
*****
use
return
```

```

*****
*
*           I N I T V A R . P R G
*-----*
*
*  TYPE:   Program Library
*
*  CALLED FROM:  PIG, bthcert, mrgcert, dthcert
*
*  GLOBAL VARIABLES:  This routine establishes the
*                    global variables used in PIG
*
*  LOGIC:  This routine initializes and establishes
*          the type, of the variables associated with
*          PERSON data base fields, as well as the
*          source number variable associated with the
*          SOURCE data base file. Global variables
*          declared here also.
*
*****
PUBLIC dte, city, cnty, st, fac, sex, rece, prsno
PUBLIC fneme, mneme, lneme, age
PUBLIC srcno, src, sfdte, sedte, sfcity, sfcnty, intfd
PUBLIC sfst, sprsno, rel, prsno, srcno
PUBLIC inerr, person, cnt, opt, birthdte
PUBLIC bdte, bcity, bcnty, bst, ceus, bfec, ddte, dcity
PUBLIC dcnty, dst, sfname, smname, slname, mfname, mmname
PUBLIC mlneme, fneme, fneme, flname, bege, bsex, brece
PUBLIC prsno2, prsno3, prsno4, prsno5, prsno6, prsno7
PUBLIC prsno8, f, m, flname, fneme, flname, mlneme
PUBLIC mfname, mmname, bfname, bmname, blname, fbname
PUBLIC fbname, fbname, mbfname, mbmname, mblname, mdte
**
  use PERSON.dbf
  eppend BLANK
  store PERS_NO to prsno
  store PERS_NO to prsno2
  store PERS_NO to prsno3
  store PERS_NO to prsno4
  store PERS_NO to prsno5
  store PERS_NO to prsno6
  store PERS_NO to prsno7
  store PERS_NO to prsno8
*****
  use SOURCE.dbf
  eppend BLANK
  store SRC_NO to srcno
*****
use
return

```

```

*****
*
*           I N I T S C . P R G
*
*-----*
*  TYPE:  Program Library
*  CALLED FROM:  pig, bthcert, mrgcert, dthcert
*
*-----*
*  GLOBAL VARIABLES:  src, sfdte, intfd, sedte,
*                    sfcity, sfcnty, sfst
*
*-----*
*  LOGIC:  This routine initializes and establishes
*          the type, of the variables associated with
*          SOURCE data base fields.
*
*****
  use SOURCE.dbf
  append BLANK
  store SRC_TYPE to src
  store DATE_SR_FD to sfdte
  store INT_FD to intfd
  store DATE_SR_ET to sedte
  store CITY_SR_FD to sfcity
  store CNTY_SR_FD to sfcnty
  store ST_SR_FD to sfst
use
return

```

```
*****
*
*           H E L P I N F O . P R G
*-----*
*   TYPE:  Program Library
*
*   CALLED FROM:  MMENU
*-----*
*
*   PROGRAMS INVOKED:  none
*
*   GLOBAL VARIABLES:  none
*-----*
*
*   LOGIC:  This procedure will provide an on-line
*           tutorial for the Professional Information
*           Genealogical System.  A detailed descrip-
*           tion will be provided.
*
*-----*
clear
@ 5,5 say "helpinfo program not implemented yet"
return
```

```

*****
*
*                               E N T E R S R C . P R G                               *
*-----*
*  TYPE:  program library
*
*  CALLED BY:  MMENU
*
*  PROGRAM INVOKED:  BTHCERT
*                   MRGCERT
*                   DTHCERT
*                   INFCERT
*
*  GLOBAL VARIABLES:  opt
*
*  LOGIC:  The user is prompted to select the source
*          document type to be entered.  The possible
*          source types for this prototype are:
*          birth, death, and marriage certificates.
*          Informal source types will be an
*          extension to be added later.
*****
clear
store " " to srctyp
do while srctyp <> "5"
  clear
  set bell off
  store space(1) to srctyp
  @ 3,19 to 6,60
  @ 7,19 to 19,60
  @ 5,30 say "SOURCE RECORD ENTRY"
  @ 9,26 say "Primary Source Record Types:"
  @ 12,26 say "(1) Birth Certificate"
  @ 13,26 say "(2) Marriage Certificate"
  @ 14,26 say "(3) Death Certificate"
  @ 15,26 say "(4) Informal Source"
  @ 16,26 say "(5) Exit - return to main menu"
  clear gets
  @ 21,28 say ;
  "Enter Source Record Type: " get srctyp
  read
do while srctyp < "1" .or. srctyp > "5"
  store " " to srctyp
  clear gets
  @ 23,28 say ;
  "Enter Source Record Type: " get srctyp
  read
enddo

```



```
*
* call appropriate procedure based on source
* type selected.
*
do case
  case srctyp = "1"
    do bthcert
  case srctyp = "2"
    do mrgcert
  case srctyp = "3"
    do dthcert
  case srctyp = "4"
```

```

*****
*
*           D E L T S R C . P R G
*-----*
*   TYPE:  Command Library
*
*   CALLED FROM:  mmenu.pgm
*-----*
*   PROGRAMS INVOKED:  dbthcert
*                     dmrgcert
*                     ddthcert
*                     dinfcert
*
*   GLOBAL VARIABLES:  opt,inerr
*-----*
*
*   LOGIC:  The user is prompted to select the type
*           of source record to be deleted.  The
*           appropriate procedure based on the source
*           record type selected is given control.
*
*****
clear
store " " to srctyp
do while srctyp <> "5"
  clear
  set bell off
  store space(1) to srctyp
  @ 3,19 to 6,60
  @ 7,19 to 19,60
  @ 5,30 say "SOURCE RECORD ENTRY"
  @ 9,26 say "Primary Source Record Types:"
  @ 12,26 say "(1) Birth Certificate"
  @ 13,26 say "(2) Marriage Certificate"
  @ 14,26 say "(3) Death Certificate"
  @ 15,26 say "(4) Informal Source"
  @ 16,26 say "(5) Exit - return to main menu"
  clear gets
  @ 21,28 say ;
  "Enter Source Record Type: " get srctyp
  read
  do while srctyp < "1" .or. srctyp > "5"
    store " " to srctyp
    clear gets
    @ 23,28 say ;
    "Enter Source Record Type: " get srctyp
    read
  enddo
  *
  * call appropriate procedure based on source
  * type selected
  *
  do case
  case srctyp = "1"
    do dbthcert
  case srctyp = "2"
    do dmrgcert
  case srctyp = "3"
    do ddthcert
  case srctyp = "4"
    do dinfcert
  case srctyp = "5"
    exit
  endcase
enddo
return

```

```

*****
*
*           V A L N A M E . P R G
*-----*
*
*   TYPE:  program library
*
*   CALLED BY:  mrgcert, bthcert, dthcert
*-----*
*
*   GLOBAL VARIABLES:  f, m, fname, mname, lname,
*                       inerr
*-----*
*
*   LOGIC:  This routine validates the name fields.
*           Required:  Last name and first and/or
*                       middle name.
*****
*
f = "N"
m = "N"
if lname <> "
    if fname = "
        if mname = "
            @ 13,4 say "No First or Middle Name Given, " ;
                "Reenter the field(s)"
        else
            inerr = "N"
            m = "Y"
        endif
    else
        inerr = "N"
        f = "Y"
    endif
else @ 13,4 say "No Last Name Provided, Reenter field"
endif
return

```

```
*****
*
*                               V A L M C . P R G                               *
*-----*
*
*   TYPE:  program library
*
*   CALLED FROM:  MRGCERT
*
*-----*
*
*   PROGRAMS INVOKED:  VALNAME
*
*-----*
*
*   LOGIC:  This program validates the marriage certi-
*           ficate.  The Last name + combination of
*           middle and/or first name is required.
*
*****
* >>>  it has to be expanded
```

```

*****
*
*                               V A L B C                               *
*-----*
* TYPE:  program library                                             *
* CALLED BY:  BTHCERT                                               *
*-----*
* GLOBAL VARIABLES:  dte, inerr, birthdate, person                 *
*-----*
* LOGIC:  This routine validates the birth certificate.*
*         A birth certificate, must have a birth *
*         date. The names on the certificate must be *
*         verified. If an error is found, the inerr *
*         is set to "Y"                                             *
*****
*
* This code needs expanded. It only has the bare
* minimum amount of checking at this point. It will
* be expanded.
*
inerr      = "n"
person     = "y"
birthdate  = "y"
* a loop needs added here to check for all names.
do valname
if dte = " / / "
    @ 20,2 say "Date of birth is required for birth cert."
    inerr = "y"
endif
return

```

```

*****
*
*                               V A L D C                               *
*-----*
* TYPE:  program library                                             *
*
* CALLED BY:  DTHCERT                                               *
*-----*
*
* GLOBAL VARIABLES:  ddte, inerr, birthdate,                        *
*                   lname, mname, fname                            *
*-----*
*
* LOGIC:  This routine validates the death certificate.*
*         A death certificate, must have a death                   *
*         date. The names on the certificate must be               *
*         verified. If an error is found, the inerr               *
*         is set to "y"                                           *
*-----*
*****
*
inerr      = "N"
birthdate = "Y"
* a loop needs added here to check for all names.
do valname
if ddte = " / / "
    @ 20,2 say "Date of death is required for death cert."
    inerr = "Y"
endif
return

```

```

*****
*
*                               V A L S R C                               *
*-----*
*
* TYPE:  program library                                             *
*
* CALLED BY:  DTHCERT, BTHCERT, MRGCERT                             *
*-----*
*
* GLOBAL VARIABLES:  sedte, intfd, inerr                             *
*-----*
*
* LOGIC:  This routine validates the source description             *
*          information.  The initials of the                         *
*          person entering the document must be given.             *
*          The date of the document must be typed in.             *
*          If an error is found, inerr is set to "Y".             *
*-----*
*****
*
inerr      = "N"
if intfd = ' '
    @ 18,2 say
        "initials of person entering source required"
    inerr = "Y"
endif
if sedte = " / / "
    @ 20,2 say "Date source entered required."
    inerr = "Y"
endif
return

```

```

*****
*                                     *
*                   B C F M T         *
*-----*
*  TYPE:  program library             *
*-----*
*  CALLED BY:  BTHCERT                *
*-----*
*  GLOBAL VARIABLES:  lname, fname, mname, dte, race, *
*                    sex, fac, city, cnty, st         *
*-----*
*  LOGIC:  This is the screen format for the birth *
*          certificate.                         *
*-----*
*****
@ 5,4 say "Name at birth:"
@ 6,4 say "Last " get lname
@ 6,29 say "First " get fname
@ 6,50 say "Middle " get mname
@ 7,4 say "Birth Date (MM/DD/YY) " get dte
@ 7,36 say "Race " get race
@ 7,53 say "Sex " get sex
@ 8,4 say "Place of birth:"
@ 9,4 say "Facility " get fac
@ 10,4 say "City " get city
@ 10,31 say "County " get cnty
@ 10,60 say "State " get st

```



```

*****
*
*                               S R C F M T
*-----*
* TYPE:  program library
*
* CALLED BY:  BTHCERT, DTHCERT, MCCERT
*-----*
*
* GLOBAL VARIABLES:  initfd, sedte, sfdte, sfcity,
*                   sfcnty, sfst
*-----*
*
* LOGIC:  This is the screen format for the source
*         record information.
*
*****
@ 13,2 say "Source Information:"
@ 14,2 say
      "Person entering source document (initials) " ;
      get initfd
@ 15,2 say "Date entered (MM/DD/YY) " get sedte
@ 15,36 say "Date source found (MM/DD/YY) " get sfdte
@ 16,2 say "Place source was found : "
@ 17,2 say "City " get sfcity
@ 17,29 say "County " get sfcnty
@ 17,58 say "State " get sfst

```

M C F M T 1

TYPE: program library

CALLED BY: MRGCERT

GLOBAL VARIABLES: lnama, fnama, mname, aex, race,
 blname, bname, bfname, flname,
 ffname, fname, mlname, mfname,
 mmname, fbname, fbfname, fbnname,
 mblname, mbfname, mbmname, baax,
 braca, бага, age

LOGIC: This is the screen format for a portion of
 the death certificate.

```

@ 4,2 say "Groom Information:."
@ 6,2 aay "Last " gat ;
      lnama picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 6,28 say "First " gat ;
      ffname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 6,50 aay "Middle " gat ;
      mname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 7,2 aay "Sex " gat aax picture "!"
@ 7,10 aay "Race " gat race picture "!"
@ 7,28 aay "Age " gat age
@ 8,2 say "Groom's Father:"
@ 9,2 aay "Last " gat ;
      flname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,28 aay "First " gat ;
      ffnname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,50 aay "Middla " gat ;
      fmname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 10,2 aay "Groom's Mother: "
@ 11,2 say "Last " gat ;
      mlname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 11,28 aay "First " gat ;
      mfnname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 11,50 aay "Middle " gat ;
      mmname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 13,2 aay "Brida Information:"
@ 15,2 say "Last " gat ;
      blname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 15,28 aay "First " gat ;
      bfnname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 15,50 aay "Middle " gat ;
      bmmname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 16,2 say "Sex " gat bsax picture "!"
@ 16,10 say "Race " gat braca picture "!"
@ 16,28 aay "Age " gat бага
@ 17,2 aay "Brida's Father:"
@ 18,2 say "Last " gat ;
    
```

```

e 18,28 say "fbname picture "!!!!!!!!!!!!!!!!!!!!!!"
          fbname picture "First " gat ;
e 18,50 say "Middla " gat ;
          fbname picture "!!!!!!!!!!!!!!!!!!!!!!"
          fbname picture "Middle " gat ;
e 19,2 say "Brida's Mothar (maidan name):"
          fbname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 20,2 say "Last " gat ;
          mblname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 20,28 say "First " gat ;
          mblname picture "!!!!!!!!!!!!!!!!!!!!!!"
          mblname picture "First " gat ;
e 20,50 say "Middle " gat ;
          mblname picture "!!!!!!!!!!!!!!!!!!!!!!"
          mblname picture "Middle " gat ;
          mblname picture "!!!!!!!!!!!!!!!!!!!!!!"

```

```

*****
*
*                               M C F M T 2
*-----*
*
* TYPE: program library
*
* CALLED BY: MRGCERT
*-----*
*
* GLOBAL VARIABLES: mdate, fac, city, cnty, st,
*                  w1lname, w1fname, w1mname,
*                  w2lname, w2fname, w2mname
*-----*
*
* LOGIC: This is the second screen format of the
*        marriage certificate entry screen routine.
*-----*
e 2,2 say "Date of Marriage (MM/DD/YY) " ;
      gat mdate
e 4,2 say "Place of Marriage:"
e 5,8 say "Facility " gat ;
      fac picture "!!!!!!!!!!!!!!!!!!!!!!"
e 6,8 say "City " gat ;
      city picture "!!!!!!!!!!!!!!!!!!!!!!"
e 6,36 say "County " gat ;
      cnty picture "!!!!!!!!!!!!!!!!!!!!!!"
e 6,69 say "State " gat st "!"
e 8,2 say "Witnesses:"
e 9,2 say "Last " gat ;
      w1lname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 9,28 say "First " gat ;
      w1fname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 9,50 say "Middle " gat ;
      w1mname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 9,2 say "Last " gat ;
      w2lname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 9,28 say "First " gat ;
      w2fname picture "!!!!!!!!!!!!!!!!!!!!!!"
e 9,50 say "Middle " gat ;
      w2mname picture "!!!!!!!!!!!!!!!!!!!!!!"

```

```

*****
*
*                               D C F M T 1
*-----*
* TYPE:  program library
*
* CALLED BY:  DTHCERT
*-----*
* GLOBAL VARIABLES:  lnama, fnama, mnama, ddte, dcity,
*                   dcnty, dst, sex, raza, caus,
*                   bdte, bcity, bst, fac, dte, city,
*                   cnty, st
*-----*
* LOGIC:  This is the scraan format for a portion of
*         the death certificate.
*
*****
@ 4,2 say "Information on the Dacaasad:"
@ 6,2 say "Laet " gat ;
           lnama picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 6,28 say "First " gat ;
           fnama picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 6,50 say "Middle " gat ;
           mnama picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 8,2 say "Data of Death (MM/DD/YY) " gat ddta
@ 9,2 say "Placa of Daath City " gat ;
           dcity picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 9,43 say " county " gat
           dcnty picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 9,71 say " Stata " gat dst picture "!!"
@ 10,2 say "Sax " gat sex picture "!!"
@ 10,10 say "Race " gat raza picture "!!!!!!!!!!!!!"
@ 10,28 say "Causa of Daath " gat ;
           caue picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 12,2 say "Date of Burial " gat bdte
@ 13,2 say "Burial Sita: City " gat ;
           bcity picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 13,41 say "county " gat ;
           bcnty picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 13,69 say "State " gat bst picture "!!"
@ 14,2 say "Burial Facility "
           gat bfac picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 16,2 say "Data of Birth " gat dte
@ 17,2 say "Place of Birth: City " gat ;
           city picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 17,44 say "County " gat ;
           cnty picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 17,72 say "Stata " gat st picture "!!"

```

```

*****
*
*                               D C F M T 2
*-----*
* TYPE:  program library
*
* CALLED BY:  DTHCERT
*-----*
*
* GLOBAL VARIABLES:  lname, fname, mname, ddate, dcity,
*                   dcnty, dst, sex, race, caus,
*                   bdte, bcity, bst, fac, dte, city,
*                   cnty, st
*-----*
*
* LOGIC:  This is the screen format for a portion of
*         the death certificate.
*-----*
*****
@ 1,1 clear
@ 1,2 say "Deceased:"
@ 2,3 say "Last ", ;
      lname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 2,28 say "First ", fname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 2,50 say "Middle ", mname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 4,2 say "Spouse: "
@ 5,2 say "Last " get ;
      slname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 5,28 say "First " get ;
      sfname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 5,50 say "Middle " get ;
      smname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 7,2 say "Father: "
@ 8,2 say "Last " get ;
      flname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 8,28 say "First " get ;
      ffname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 8,50 say "Middle " get ;
      fmmname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 10,2 say "Mother: "
@ 11,2 say "Last " get ;
      mlname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 11,28 say "First " get ;
      mfname picture "!!!!!!!!!!!!!!!!!!!!!!!"
@ 11,50 say "Middle " get ;
      mmname picture "!!!!!!!!!!!!!!!!!!!!!!!"

```

```

*****
*
*           D T H C E R T . P R G
*-----*
*   TYPE: Program Library
*
*   CALLED FROM: ENTERSRC
*-----*
*   PROGRAMS INVOKED: DCFMT1, DCFMT2, SRCFMT,
*                   VALDC, VALSRC, CRTSRCNO,
*                   CRTPRSNO, ADDSRC, ADDDC,
*                   ADDBC, ADDPRS, ADDMC,
*                   INITSC, INITSCRV
*
*   GLOBAL VARIABLES: cnt, inerr, rel, opt, sex,
*                   fname, mname, lname, mfname,
*                   mmname, mlname, sfname, smname,
*                   slname
*-----*
*
*   LOGIC: The death certificate screen(s) is printed.
*          The user is prompted to enter the death
*          certificate data. The data is validated.
*          If errors are found, the user is prompted
*          to reenter the fields in error. If no
*          errors, the source data is added to the
*          appropriate data base files.
*-----*
*****
clear
src = "DC"
cnt = 1
loopcnt1 = "Y"
opt = " "

@ 1,32 say "ENTER DEATH CERTIFICATE"
do while loopcnt1 = "Y"
do dcfmt1
@ 20,1 say " "
wait
@ 1,1 clear
do dcfmt2
do srcfmt
@ 21,2 say ;
"Type X to Exit, any other key to Enter Data";
get opt picture "!"
read
if opt = "X"
return
endif
do valdc
do valsrc
if inerr = "Y"
@ 22,2 say "Reenter the field(s) in error"
endif
loopcnt1 = inerr

```

```

@ 1,1 clara
enddo
*
* Is this a new person, or does this person exist?
*
* >>>>>> code goes here

* Enter death certificate information into db's
* Assumption for this section of code is that the
* people are new. This section will have to be
* modified.
*
cnt = 1
do while cnt < 5

  * Add deceased

  casa cnt = 1 .and. lnama <> " "
  do crtsrcno
  do crtprsrno
  do addprs
  do addsrc
  do adddc
  if birthdate = "Y"
    do addbc
  endif

  ** add deceased spouse

  casa cnt = 2 .and. slnama <> " "
  do crtprsrno
  do addsrc
  do addmc
  fname = sfname
  mname = smname
  lnama = slnama
  do addprs
  if sax = "F"
    ral = "HUSBAND"
  else
    rel = "WIFE "
  endif
  do addral

  ** add deceased father

  casa cnt = 3 .and. flnama <> " "
  do crtprsrno
  do addsrc
  fname = ffname
  mname = fmname
  lnama = flnama
  do addprs
  ral = "FATHER"
  do addrel

  ** add deceased's mother

  casa cnt = 4 .and. mlnama <> " "
  do crtprsrno

```

```
do adderc
  fname = #fname
  mname = #mname
  lname = #lname
  do addprs
    rel = "MOTHER"
  do addel
  endcase
  cnt = cnt + 1
enddo
use
do initsc
do initscr
return
```



```

*****
*
*           B T H C E R T . P R G
*-----*
*   TYPE:  Program Library
*
*   CALLED FROM:  ENTERSRC
*-----*
*   PROGRAMS INVOKED:  BCFMT, SRCFMT,
*                     VALBC, VALSRC, CRTSRCNO,
*                     CRTPRSNO, ADDSRC, ADDPRS,
*                     ADDBC, INITSC, INITSCRV
*
*   GLOBAL VARIABLES:  cnt, inerr, rel, opt,
*                     birthdate, person
*-----*
*   LOGIC:  The birth certificate screen(s) is printed.
*           The user is prompted to enter the birth
*           certificate data. The data is validated.
*           If errors are found, the user is prompted
*           to reenter the fields in error. If no
*           errors, the source data is added to the
*           appropriate data base files.
*-----*
*****
clear
src = "BC"
cnt = 1
loopcntl = "y"
opt = " "

@ 1,32 say "ENTER BIRTH CERTIFICATE"

do while cnt < 4
@ 3,2 clear
do while loopcntl = "y"
do case
case cnt = 1
@ 3,2 say ;
"Infant Birth Certificate Information:"
case cnt = 2
@ 3,2 say ;
"Father Birth Certificate Information:"
case cnt = 3
@ 3,2 say ;
"Mother Birth Certificate Information:"
endcase
do bcfmt
if cnt = 1
do srcfmt
endif
@ 21,2 say ;
"Type X to Exit, any other key to enter data";
get opt
read
if opt = "X"
return

```

```

endif
do valbc
if inerr = "y"
@ 22,2 say "Reenter the field(s) in error"
endif
loopcntl = inerr
enddo
*
* >> Is this a new person, or does this person exist?
* This code will be added later. First testing assumes
* that all individuals are new. The following code
* will have to be modified when this new code is added.
*
if cnt = 1
do crtsrcno
endif
if person = "y"
do crtprsno
if cnt <> 1
src = "BR"
endif
do addsrc
do addprs
if birthdats = "y"
do addbc
endif
do case
case cnt = 1
rel = "self"
case cnt = 2
rel = "father"
case cnt = 3
rel = "mother"
endcase
do addrel
endif
cnt = cnt + 1
loopcntl = "y"
do initscrv
enddo
do initsc
uss
return

```

```

*****
*                                     *
*                               M R G C E R T . P R G                               *
*-----*
* TYPE: Program Library *
* CALLED FROM: ENTERSRC *
*-----*
* PROGRAMS INVOKED: MCFMT1, MCFMT2, SRCFMT, *
*                   VALMC, VALSRC, CRTSRCNO, *
*                   CRTFRSNO, ADDSRC, ADDMC, *
*                   ADDFRS, INITSC, INITSCRV *
*                   INITSC, INITSCRV *
*
* GLOBAL VARIABLES: cnt, inerr, rel, opt, sex, beex, *
*                   fname, mname, lname, bfname, *
*                   bmname, blname, mfname, mmname, *
*                   mlname, ffname, fmname, flname, *
*                   fbname, fbmname, fblname, *
*                   mbfname, mbmname, mblname, *
*                   wlfname, wlfname, wllname, *
*                   w2fname, w2mname, w2lname *
*-----*
*
* LOGIC: The marriage certificate screen is printed. *
*         The user is prompted to enter the marriage *
*         certificate data. The data is validated. *
*         If errors are found, the user is prompted *
*         to reenter the fields in error. If no *
*         errors, the source data is added to the *
*         appropriate data base files. *
*****
clear
src = "MC"
cnt = 1
loopcntl = "Y"
opt = " "

@ 1,30 say "ENTER MARRIAGE DEATH CERTIFICATE"

do while loopcntl = "Y"
do mcfmt1
@ 20,1 say " "
wait
@ 1,1 clear
do mcfmt2
do srcfmt
@ 21,2 say ;
"Type X to Exit, any other key to Enter Data";
get opt picture "!"
read
if opt = "X"
return
endif
do valmc
do valsrc

```

```

if inerr = "y"
  @ 22,2 say "Reenter the field(s) in error"
endif
loopcnt1 = inerr
@ 1,1 clear
enddo
*
* Is this a new person, or does this person exist?
*
* >>>>>> code goes here
*
* Enter death certificate information into db's
* This code assumes that all of the individual's are
* new. It will be modified when the "does this person
* exist" code is added.
*
cnt = 1
do while cnt < 9
  do case

    ** Add groom

    case cnt = 1 .and. lname <> " "
      do crtprcno
      do crtprcno
      do addprs
      do addsrc
      do addmc
      rel = "SELF"
      do addrel

    ** Add bride

    case cnt = 2 .and. blname <> " "
      do crtprcno
      do addsrc
      fname = bfname
      mname = bmname
      lname = blname
      addprs
      do addmc
      rel = "WIFE"
      do addrel

    ** Add groom's father

    case cnt = 3 .and. flname <> " "
      do crtprcno
      do addsrc
      fname = ffname
      mname = ffname
      lname = flname
      addprs
      do addmc
      rel = "FATHER"
      do addrel

    ** Add groom's mother

    case cnt = 4 .and. mlname <> " "

```

```

do crtprsno
do addsrc
fname = mfname
mnama = mmnama
lname = mlnama
addprs
do addmc
rel = "MOTHER"
do addrel

** Add Brida's fathar

case cnt = 5 .and. fblname = "
prsno = prsno2
do crtprsno
do addsrc
fname = fbfname
mnama = fbmnama
lname = fblname
addprs
do addmc
rel = "FATHER"
do addrel

** Add Bride's mother

casa cnt = 6 .and. mblname = "
prsno = prsno2
do crtprsno
do addsrc
fname = mbfname
mnama = mbmnama
lname = mblname
do addprs
do addmc
rel = "MOTHER"
do addrel

** Add Witness 1

case cnt = 7 .and. wllname = "
do crtprsno
do addsrc
fname = wlfname
mnama = wlmnama
lname = wllname
addprs
do addmc
rel = "WITNESS"
do addrel

** Add Witness 2

case cnt = 8 .and. w2lnama = "
do crtprsno
do addsrc
fname = w2fname
mnama = w2mnama
lname = w2lnama
addprs
do addmc
rel = "WIFE"
do addrel
andcasa
cnt = cnt + 1
anddo
usa
do initsc
do initscrv
return

```

```

*****
*
*                               DBTHCERT.PRG
*-----*
*   CALLED FROM:  DELTSRC.PRG
*
*   LOGIC:        THIS IS THE PROGRAM THAT HANDLES THE LOGIC
*                  REQUIRED TO DELETE A BIRTH CERTIFICATE.
*                  THE USER SUPPLIES THE NAME FOR THE PRIMARY
*                  PERSON ON THE BC THEN THE PROGRAM VALIDATES
*                  THE NAME AND SEARCHES FOR THE APPROPRIATE
*                  NAME COMBINATION.  THE PROGRAM PROVIDES A
*                  LIST OF ALL NAMES AND THEIR ASSOCIATED
*                  PERSON NUMBERS AND SOURCE NUMBERS THAT
*                  MATCH WHAT THE USER SPECIFIED.  THE USER
*                  THEN SELECTS THE APPROPRIATE PERSON AND
*                  SOURCE.  THIS PROGRAM THEN INVOKES THE
*                  PROGRAMS REQUIRED TO DELETE DATA FROM ALL
*                  DATABASE FILES REQUIRED.
*-----*
*
*   PROGRAMS INVOKED: VALNAME.PRG, DELTPERS, DELTSRCE,
*                   DELTBC, DELTREL, INITSRCV, INITSC
*-----*
*
*   ACTIVE DATABASE FILES:  BIRTH_IN.DBF, SOURCE.DBF,
*                           PERSON.DBF
*
*   ACTIVE INDEX FILES:    BIRTH_IN -> SRCPERS.NDX
*                           SOURCE -> PERSTYPE.NDX, SPERS.NDX
*                           PERSON -> NAME.NDX, NAME2.NDX,
*                               NAME3.NDX
*-----*
*
*   INITIALIZE VARIABLES
*
clear
opt = " "
pno = " "
match = "y"
stype = "BC"

*
*   ESTABLISH ACTIVE WORK AREAS
*   WITH APPROPRIATE DATABASE FILES
*   AND THE REQUIRED INDEX FILES
*
*   ALSO, ESTABLISH RELATIONS
*
select 3
use BIRTH_IN index SRCPERS alias BIRTH
select 2
use SOURCE index PERSTYPE,SPERS alias SRCE
set relation to SRC_NO+PERS_NO into BIRTH
select 1

```

```
use PERSON index NAME,NAME2,NAME3 alias PEOPLE
set relation to PERS_NO+"BC" into SRCE
```

```
*
* GET PRIMARY PERSON'S NAME AND VALIDATE *
*
```

```
loopcnt1 = "Y"
do while loopcnt1 = "Y"
  inerr = "Y"
  do while inerr = "Y"
    @ 5,2 say "Enter Name of Person Whose Birth Certificate"
    @ 5,47 say "is to be Deleted"
    @ 6,2 say "Required: Last Name + First and/or Middle"
    @ 7,2 say "If Last Name is all that is Known, Please"
    @ 7,44 say "Perform Query Instead of Delete."
    @ 11,2 say "LAST " get lname
    @ 11,27 say "FIRST " get fname
    @ 11,48 say "MIDDLE " get mname
    @ 21,2 say "TYPE X to Exit, Any Other Key to Enter Data ";
    get opt
    read
    if opt = "X"
      return
    endif
    do valname
  enddo while inerr
```

```
*
* Search For Name In Format Provided By User *
*
```

```
if f = "Y" .and. m = "Y"
  set order to 1
  seek ltrim(trim(lname))+ltrim(trim(fname))+ltrim(trim(mname))
else
  if f = "Y" .and. m = "N"
    set order to 2
    seek ltrim(trim(lname))+ltrim(trim(fname))
  else
    set order to 3
    seek ltrim(trim(lname))+ltrim(trim(mname))
  endif
endif
if eof()
  @ 14,4 say "Match Can Not Be Found For Name Entered"
  loopcnt1 = "Y"
else
```

```
*
* CHECK TO SEE IF PERSON ENTERED HAS A BC *
*
```

```
loopcnt1 = "N"
prtcnt = 0
do while match = "Y"
  select 2
  if .not. eof()
```

```

* PRINT ALL PERTINENT DATA TO SCREEN *
*
clear
@ prtcnt+1,1 say PERS_NO->PERSON,PR_L_NAME->PERSON,;
PR_F_NAME->PERSON,PR_M_NAME->PERSON
@ prtcnt+1,60 say BIRTH_DATE->BIRTH_IN,SR_NO,INT_FD,;
ST_SR_FD
prtcnt = prtcnt + 1
endif

* SEARCH FOR ANOTHER PERSON WITH *
* SAME NAME AS ENTERED BY THE USER *
*

select 1
skip
person = "Y"

* DETERMINE WHICH NAME COMBINATION *
* USER PROVIDED *
*

if .not. eof()
if f="Y" .and. m="Y"
if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
ltrim(trim(PR_F_NAME))<>ltrim(trim(fname)) .or. ;
ltrim(trim(PR_M_NAME))<>ltrim(trim(mname))
person = "N"
endif
else
if f="Y" .and. m="N"
set order to 2
if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
ltrim(trim(PR_F_NAME))<>ltrim(trim(fname))
person = "N"
endif
else
set order to 3
if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
ltrim(trim(PR_M_NAME))<>ltrim(trim(mname))
person = "N"
endif
endif
endif
else
person = "N"
endif

*
* CHECK FOR LAST PERSON THAT *
* MATCHES NAME ENTERED *
*

if person = "N"

*
* CHECK TO SEE IF ANY LEGAL *
* ENTRIES WERE ENCOUNTERED *

```



```

if prtcnt = 0
  @ 10,2 say "Person Does Not Have A Birth "
  @ 10,31 say "Certificate Entry"
  @ 21,4 say "Type Any Key to Return" get opt
  read
  return
else
  *
  * HAVE ALL MATCHES - ASK USER *
  * TO CHOOSE APPROPRIATE ONE *
  *
  @ 20,4 say "Enter: Person # ",get prsno," Source #",;
  get srcno
  read
  match = "N"
endif
endif
enddo while match
endif

*
* DOES THE PERSON SELECTED HAVE OTHER *
* SOURCE RECORDS? IF SO, DON'T DELETE *
* PERSON; IF NOT, DELETE PERSON *
*
select 2
set order to 2
seek prsno
if SRC_NO = srcno
  skip
  if eof() .or. PERS_NO = pno
    use
    do DELTPERS
  endif
endif
use
*
* DELETE SOURCE INFO *
*
do DELTSRCE
*
* DELETE BIRTH INFO *
*
do DELTBC
*
* DELETE RELATION INFO *
*
do DELTREL
*
* CLEAR SCREENS AND VARIABLES *
*
do INITSCRV
do INITSC
return

```

```

*****
*                               DMRGCERT.PRG                               *
*-----*
* CALLED FROM: DELTSRC.PRG                                             *
*
* LOGIC: THIS IS THE PROGRAM THAT HANDLES THE LOGIC REQUIRED TO DELETE A MARRIAGE CERTIFICATE.
* THE USER SUPPLIES THE NAME FOR THE PRIMARY PERSON ON THE MC THEN THE PROGRAM VALIDATES
* THE NAME AND SEARCHES FOR THE APPROPRIATE NAME COMBINATION. THE PROGRAM PROVIDES A
* LIST OF ALL NAMES AND THEIR ASSOCIATED PERSON NUMBERS AND SOURCE NUMBERS THAT
* MATCH WHAT THE USER SPECIFIED. THE USER THEN SELECTS THE APPROPRIATE PERSON AND
* SOURCE. THIS PROGRAM THEN INVOKES THE PROGRAMS REQUIRED TO DELETE DATA FROM ALL
* DATABASE FILES REQUIRED.
*-----*
* PROGRAMS INVOKED: VALNAME.PRG, DELTPERS, DELTSRCE, DELTMC, DELTREL, INITSCRV, INITSC
*-----*
* ACTIVE DATABASE FILES: MARRIAGE.DBF, SOURCE.DBF, PERSON.DBF
*
* ACTIVE INDEX FILES: MARRIAGE -> MSRCPERS.NDX
* SOURCE -> PERSTYPE.NDX, SPERS.NDX
* PERSON -> NAME.NDX, NAME2.NDX, NAME3.NDX
*-----*
*
* INITIALIZE VARIABLES
*
clear
opt = " "
pno = " "
match = "Y"
stype = "MC"

* ESTABLISH ACTIVE WORK AREAS
* WITH APPROPRIATE DATABASE FILES
* AND THE REQUIRED INDEX FILES
* ALSO, ESTABLISH RELATIONS

select 3
use MARRIAGE index MSRCPERS alias MARRY
select 2
use SOURCE index PERSTYPE,SPERS alias SRCE
set relation to SRC_NO+PERS_NO into MARRY
select 1

```

```
use PERSON index NAME,NAME2,NAME3 alias PEOPLE
set relation to PERS_NO+"MC" into SRCE
```

```
*
* GET PRIMARY PERSON'S NAME AND VALIDATE *
*
```

```
loopcntl = "Y"
do while loopcntl = "Y"
  inerr = "Y"
  do while inerr = "Y"
    @ 5,2 say "Enter Name of Person Whose Marriage"
    @ 5,47 say "Certificate is to be Deleted"
    @ 6,2 say "Required: Last Name + First and/or Middle"
    @ 7,2 say "If Last Name is all that is Known, Please"
    @ 7,44 say "Perform Query Instead of Delete."
    @ 11,2 say "LAST " get lname
    @ 11,27 say "FIRST " get fname
    @ 11,48 say "MIDDLE " get mname
    @ 21,2 say "TYPE X to Exit, Any Other Key to Enter Data ";
    get opt
    read
    if opt = "X"
      return
    endif
    do valname
  enddo while inerr
```

```
*
* Search For Name In Format Provided By User *
*
```

```
if f = "Y" .and. m = "Y"
  set order to 1
  seek ltrim(trim(lname))+ltrim(trim(fname))+ltrim(trim(mname))
else
  if f = "Y" .and. m = "N"
    set order to 2
    seek ltrim(trim(lname))+ltrim(trim(fname))
  else
    set order to 3
    seek ltrim(trim(lname))+ltrim(trim(mname))
  endif
endif
if eof()
  @ 14,4 say "Match Can Not Be Found For Name Entered"
  loopcntl = "Y"
else
```

```
*
* CHECK TO SEE IF PERSON ENTERED HAS A MC *
*
```

```
loopcntl = "N"
prtcnt = 0
do while match = "Y"
  select 2
  if .not. eof()
```

```

*   PRINT ALL PERTINENT DATA TO SCREEN   *
*
clear
@ prtcnt+1,1 say PERS NO->PERSON,PR L NAME->PERSON,;
PR F NAME->PERSON,PR M NAME->PERSON
@ prtcnt+1,60 say MARRY DATE->MARRIAGE,SRC_NO,INT_FD,;
ST_SR_FD
prtcnt = prtcnt + 1
endif

*
*   SEARCH FOR ANOTHER PERSON WITH       *
*   SAME NAME AS ENTERED BY THE USER    *
*
select 1
skip
person = "Y"

*   DETERMINE WHICH NAME COMBINATION     *
*   USER PROVIDED                       *
*
if .not. eof()
  if f="Y" .and. m="Y"
    if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
       ltrim(trim(PR_F_NAME))<>ltrim(trim(fname)) .or. ;
       ltrim(trim(PR_M_NAME))<>ltrim(trim(mname))
      person = "N"
    endif
  else
    if f="Y" .and. m="N"
      set order to 2
      if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
         ltrim(trim(PR_F_NAME))<>ltrim(trim(fname))
        person = "N"
      endif
    else
      set order to 3
      if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
         ltrim(trim(PR_M_NAME))<>ltrim(trim(mname))
        person = "N"
      endif
    endif
  endif
else
  person = "N"
endif

*
*   CHECK FOR LAST PERSON THAT           *
*   MATCHES NAME ENTERED                 *
*
if person = "N"

*
*   CHECK TO SEE IF ANY LEGAL           *
*   ENTRIES WERE ENCOUNTERED           *

```

```

if prtcnt = 0
  @ 10,2 say "Person Does Not Have A Marriage"
  @ 10,31 say "Certificate Entry"
  @ 21,4 say "Type Any Key to Return" get opt
  read
  return
else
  *
  *   HAVE ALL MATCHES - ASK USER   *
  *   TO CHOOSE APPROPRIATE ONE   *
  *
  @ 20,4 say "Enter: Person # ",get prsno," Source #",;
  get srcno
  read
  match = "N"
endif
endif
enddo while match
endif

*
*   DOES THE PERSON SELECTED HAVE OTHER
*   SOURCE RECORDS? IF SO, DON'T DELETE
*   PERSON; IF NOT, DELETE PERSON
*
select 2
set order to 2
seek prsno
if SRC NO = srcno
  skip
  if eof() .or. PERS_NO = prsno
    use
    do DELTPERS
  endif
endif
use
*
*   DELETE SOURCE INFO
*
do DELTSRCE
*
*   DELETE MARRIAGE INFO
*
do DELTMC
*
*   DELETE RELATION INFO
*
do DELTREL
*   CLEAR SCREENS AND VARIABLES
*

do INITSCRV
do INITSC
return

```

```

*****
*                                     *
*                               DDTHCERT.PRG                               *
*-----*
*  CALLED FROM:  DELTSRC.PRG                                             *
*
*  LOGIC:  THIS IS THE PROGRAM THAT HANDLES THE LOGIC                   *
*          REQUIRED TO DELETE A DEATH CERTIFICATE.                       *
*          THE USER SUPPLIES THE NAME FOR THE PRIMARY                   *
*          PERSON ON THE DC THEN THE PROGRAM VALIDATES                   *
*          THE NAME AND SEARCHES FOR THE APPROPRIATE                     *
*          NAME COMBINATION.  THE PROGRAM PROVIDES A                     *
*          LIST OF ALL NAMES AND THEIR ASSOCIATED                       *
*          PERSON NUMBERS AND SOURCE NUMBERS THAT                       *
*          MATCH WHAT THE USER SPECIFIED.  THE USER                     *
*          THEN SELECTS THE APPROPRIATE PERSON AND                       *
*          SOURCE.  THIS PROGRAM THEN INVOKES THE                       *
*          PROGRAMS REQUIRED TO DELETE DATA FROM ALL                     *
*          DATABASE FILES REQUIRED.                                       *
*-----*
*
*  PROGRAMS INVOKED: VALNAME.PRG, DELTPERS, DELTSCRC,                   *
*                  DELTDC, DELTREL, INITSCRV, INITSC                     *
*-----*
*
*  ACTIVE DATABASE FILES:  DEATH_IN.DBF, SOURCE.DBF,                   *
*                          PERSON.DBF                                   *
*
*  ACTIVE INDEX FILES:  DEATH_IN -> DSRCPERS.NDX                       *
*                      SOURCE -> PERSTYPE.NDX, SPERS.NDX             *
*                      PERSON -> NAME.NDX, NAME2.NDX,                 *
*                          NAME3.NDX                                   *
*-----*
*
*  INITIALIZE VARIABLES  *
*
clear
opt = " "
pno = " "
match = "y"
stype = "DC"

*
*  ESTABLISH ACTIVE WORK AREAS
*  WITH APPROPRIATE DATABASE FILES
*  AND THE REQUIRED INDEX FILES
*
*  ALSO, ESTABLISH RELATIONS
*

select 3
use DEATH_IN index DSRCPERS alias DEATH
select 2
use SOURCE index PERSTYPE,SPERS alias SRCE
set relation to SRC_NO+PERS_NO into DEATH
select 1

```

```
use PERSON index NAME,NAME2,NAME3 alias PEOPLE
set relation to PERS_NO+"DC" into SRCE
```

```
*
*   GET PRIMARY PERSON'S NAME AND VALIDATE   *
*
```

```
loopcntl = "Y"
do while loopcntl = "Y"
  inerr = "Y"
  do while inerr = "Y"
    @ 5,2 say "Enter Name of Person Whose Death"
    @ 5,47 say "Certificate is to be Deleted"
    @ 6,2 say "Required: Last Name + First and/or Middle"
    @ 7,2 say "If Last Name is all that is Known, Please"
    @ 7,44 say "Perform Query Instead of Delete."
    @ 11,2 say "LAST " get lname
    @ 11,27 say "FIRST " get fname
    @ 11,48 say "MIDDLE " get mname
    @ 21,2 say "TYPE X to Exit, Any Other Key to Enter Data ";
    get opt

    read
    if opt = "X"
      return
    endif
    do valname
  enddo while inerr
```

```
*
*   Search For Name In Format Provided By User   *
*
```

```
if f = "Y" .and. m = "Y"
  set order to 1
  seek ltrim(trim(lname))+ltrim(trim(fname))+ltrim(trim(mname))
else
  if f = "Y" .and. m = "N"
    set order to 2
    seek ltrim(trim(lname))+ltrim(trim(fname))
  else
    set order to 3
    seek ltrim(trim(lname))+ltrim(trim(mname))
  endif
endif
eof()
@ 14,4 say "Match Can Not Be Found For Name Entered"
loopcntl = "Y"
else
```

```
*
*   CHECK TO SEE IF PERSON ENTERED HAS A DC   *
*
```

```
loopcntl = "N"
prctcnt = 0
do while match = "Y"
  select 2
  if .not. eof()
```

```

*   PRINT ALL PERTINENT DATA TO SCREEN   *
*                                         *

clear
@ prtcnt+1,1 say PERS_NO->PERSON,PR_L_NAME->PERSON,;
                PR_F_NAME->PERSON,PR_M_NAME->PERSON
@ prtcnt+1,60 say DEATH_DATE->DEATH_IN,SRC_NO,INT_FD,;
                ST_SR_FD
prtcnt = prtcnt + 1
endif

*                                         *
*   SEARCH FOR ANOTHER PERSON WITH       *
*   SAME NAME AS ENTERED BY THE USER    *
*                                         *

select 1
skip
person = "Y"

*   DETERMINE WHICH NAME COMBINATION     *
*   USER PROVIDED                       *
*                                         *

if .not. eof()
  if f="Y" .and. m="Y"
    if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
      ltrim(trim(PR_F_NAME))<>ltrim(trim(fname)) .or. ;
      ltrim(trim(PR_M_NAME))<>ltrim(trim(mname))
    person = "N"
  endif
else
  if f="Y" .and. m="N"
    set order to 2
    if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
      ltrim(trim(PR_F_NAME))<>ltrim(trim(fname))
    person = "N"
  endif
else
  set order to 3
  if ltrim(trim(PR_L_NAME))<>ltrim(trim(lname)) .or.;
    ltrim(trim(PR_M_NAME))<>ltrim(trim(mname))
  person = "N"
  endif
endif
endif
else
  person = "N"
endif

*                                         *
*   CHECK FOR LAST PERSON THAT          *
*   MATCHES NAME ENTERED                *
*                                         *

if person = "N"

*                                         *
*   CHECK TO SEE IF ANY LEGAL          *
*   ENTRIES WERE ENCOUNTERED          *

```



```

if prtcnt = 0
  @ 10,2 say "Person Does Not Have A Death"
  @ 10,31 say "Certificate Entry"
  @ 21,4 say "Type Any Key to Return" get opt
  read
  return
else
  *
  *   HAVE ALL MATCHES - ASK USER   *
  *   TO CHOOSE APPROPRIATE ONE   *
  *
  @ 20,4 say "Enter: Person # ",get prsno," Source #",;
  get srcno
  read
  match = "N"
endif
enddo while match
endif

*
*   DOES THE PERSON SELECTED HAVE OTHER   *
*   SOURCE RECORDS? IF SO, DON'T DELETE   *
*   PERSON; IF NOT, DELETE PERSON   *
*
select 2
set order to 2
seek prsno
if SRC_NO = srcno
  skip
  if eof() .or. PERS_NO = prsno
    use
    do DELTPERS
  endif
endif
use
*
*   DELETE SOURCE INFO   *
*
do DELTSRCE
*
*   DELETE DEATH INFO   *
*
do DELTDC
*
*   DELETE RELATION INFO   *
*
do DELTREL
*   CLEAR SCREENS AND VARIABLES   *
*
do INITSCRV
do INITSC
return

```

```

*****
*
*                               DELTPERS.PRG
*-----*
*
*   INVOKED BY:  DBTHCERT.PRG, DMRGCERT.PRG,
*                DDTHCERT.PRG
*
*   LOGIC:      THIS PROGRAM PROVIDES FOR THE DELETE
*                OF A PERSON FROM THE PERSON DATABASE
*
*                ALL INDEX FILES ASSOCIATED WITH
*                PERSON ARE RE-INDEXED TO ENSURE
*                DATA INTEGRITY,
*-----*
*
*   PROGRAMS INVOKED:  NONE
*
*   GLOBAL VARIABLES:  prsno
*-----*
*
*   ACTIVE DATABASE FILES:  PERSON.DBF
*
*   ACTIVE INDEX FILES:    PERS.NDX, NAME.NDX,
*                          NAME2.NDX, NAME3.NDX
*-----*
*****

```

```

use PERSON index PERS,NAME,NAME2,NAME3
delete for PERS_NO = prsno
pack
reindex
use
return

```

```

*****
*                                     DELTSRCE.PRG                                     *
*-----*
* INVOKED BY:  DBTHCERT.PRG, DMRGCERT.PRG, DDTHCERT.PRG *
* LOGIC:      THIS PROGRAM PROVIDES THE ACTUAL DELETE *
*             OF ALL SOURCE RECORD ENTRIES FOR A *
*             PARTICULAR SOURCE RECORD.  ENTRIES FOR *
*             PRIMARY AND SECONDARY PEOPLE ARE DELETED *
*             ALL ACTIVE INDEX FILES ARE RE-INDEXED TO *
*             ENSURE DATA INTEGRITY. *
*-----*
* PROGRAMS INVOKED:  NONE *
* GLOBAL VARIABLES:  srcno *
*-----*
* ACTIVE DATABASE FILES:  SOURCE.DBF *
* ACTIVE INDEX FILES:    SRC.NDX, PERSTYPE.NDX, *
*                       SPERS.NDX *
*****
use SOURCE index SRC,PERSTYPE,SPERS
delete for SRC_NO = srcno
pack
reindex
use
return

```

```

*****
*                               DELTREL.PRG                               *
*-----*
* INVOKED BY:  DBTHCERT.PRG, DMRGCERT.PRG, *
*              DDTHCERT.PRG              *
* LOGIC:      THIS PROGRAM PROVIDES THE ACTUAL *
*              DELETE OF ALL RELATIONSHIPS *
*              CREATED FROM A PARTICULAR SOURCE *
*              RECORD.                      *
*              ALL ACTIVE INDEX FILES ARE ALSO *
*              RE-INDEXED TO ENSURE DATA INTEGRITY *
*-----*
* PROGRAMS INVOKED:  NONE *
* GOLBAL VARIABLES:  srcno *
*-----*
* ACTIVE DATABASE FILES:  RELATION.DBF *
* ACTIVE INDEX FILES:    RSRC.NDX *
*****

```

```

use RELATION index RSRC
delete for SRC_NO = srcno
pack
reindex
use
return

```

```

*****
*                               DELTBC.PRG                               *
*-----*
*                               *
* INVOKED BY:   DBTHCERT.PRG,   *
*                               *
* LOGIC:        THIS PROGRAM PROVIDES THE ACTUAL DELETE*
*               OF ALL BIRTH INFORMATION ASSOCIATED *
*               WITH A PARTICULAR SOURCE RECORD      *
*-----*
*                               *
* PROGRAMS INVOKED:  NONE      *
*                               *
* GLOBAL VARIABLES:  srcno     *
*-----*
*                               *
* ACTIVE DATABASE FILES:  BIRTH_IN.DBF                *
*                               *
* ACTIVE INDEX FILES:    BSRC.NDX, SRCPERS.NDX        *
*-----*
*****

```

```

use BIRTH_IN index BSRC,SRCPERS
delete for SRC_NO = srcno
pack
reindex
use
return

```

```

*****
*                               DELTMC.PRG                               *
*-----*
* INVOKED BY:  DMRGCERT.PRG                                           *
* LOGIC:       THIS PROGRAM PROVIDES THE ACTUAL                       *
*              DELETE OF MARRIAGE INFORAMTION                       *
*              ASSOCIATED WITH A PARTICULAR                          *
*              SOURCE RECORD.                                         *
*              ALSO, ALL ACTIVE INDEX FILES ARE                      *
*              REINDEXED TO ENSURE DATA INTEGRITY                   *
*-----*
* PROGRAMS INVOKED:  NONE                                             *
* GLOBAL VARIABLES:  srcno                                           *
*-----*
* ACTIVE DATABASE FILES:  MARRIAGE.DBF                               *
* ACTIVE INDEX FILES:    MSRC.NDX, MSRCPERS.NDX                       *
*****

```

```

use MARRIAGE index MSRC,MSRCPERS
delete for SRC_NO = srcno
pack
reindex
use
return

```

```

*****
*                               DELTDC.PRG                               *
*-----*
*                               *
* INVOKED BY:  DDTHCERT.PRG    *
*                               *
* LOGIC:       THIS PROGRAM PROVIDES THE ACTUAL    *
*              DELETE OF DEATH INFORMATION        *
*              ASSOCIATED WITH A PARTICULAR      *
*              SOURCE RECORD                     *
*                               *
*              ALSO, ALL ACTIVE INDEX FILES ARE  *
*              RE-INDEXED TO ENSURE DATA INTEGRITY *
*-----*
*                               *
* PROGRAMS INVOKED:  NONE    *
*                               *
* GLOBAL VARIABLES:  srcno   *
*                               *
*-----*
*                               *
* ACTIVE DATABASE FILES:  DEATH_IN.DBF          *
*                               *
* ACTIVE INDEX FILES:    DSRC.NDX, DSRCPERS.NDX *
*                               *
*****

```

```

use DEATH_IN index DSRC,DSRCPERS
delete for SRC_NO = srcno
pack
reindex
use
return

```

```

*****
*                                     *
*                               ADDPRS *
*-----*
*
*   INVOKED BY:  BTHCERT.PRG, MRGCERT.PRG *
*               DTHCERT.PRG             *
*
*   LOGIC:      THIS PROGRAMS PROVIDES THE *
*               ACTUAL ADDITION OF A PERSON *
*               TO THE PERSON DATABASE.    *
*-----*
*
*   PROGRAMS INVOKED:  NONE                *
*
*   GLOBAL VARIABLES:  prsno, prsno2, prsno3, *
*                     prsno4, prsno5, prsno6 *
*                     prsno7, prsno8, fname, *
*                     mname, lname          *
*-----*
*
*   ACTIVE DATABASE FILES:  PERSON.DBF     *
*
*   ACTIVE INDEX FILES:    PERS.NDX, NAME.NDX, *
*                           NAME2.NDX, NAME3.NDX *
*-----*
*****

```

```

use PERSON.DBF index PERS,NAME,NAME2,NAME3
append blank
do case
  case cnt = 1
    replace PERS_NO with prsno
  case cnt = 2
    replace PERS_NO with prsno2
  case cnt = 3
    replace PERS_NO with prsno3
  case cnt = 4
    replace PERS_NO with prsno4
  case cnt = 5
    replace PERS_NO with prsno5
  case cnt = 6
    replace PERS_NO with prsno6
  case cnt = 7
    replace PERS_NO with prsno7
  case cnt = 8
    replace PERS_NO with prsno8

```



```
endcase
replace PR_F_NAME with fname
replace PR_M_NAME with mname
replace PR_L_NAME with lname
use
```

```

*****
*                                     ADDSRC                                     *
*-----*
* INVOKED BY:  BTHCERT.PRG, DTHCERT.PRG,                                     *
*              MRGCERT.PRG                                                 *
*
* LOGIC:       THIS PROGRAM PROVIDES THE                                     *
*              ACTUAL ADDITION OF A SOURCE                                 *
*              ENTRY INTO THE SOURCE DATABASE                               *
*-----*
* PROGRAMS INVOKED:  NONE                                                  *
*
* GLOBAL VARIABLES:  srcno, prsno, prsno2,                                  *
*                   prsno3, prsno4, prsno5,                                  *
*                   prsno6, prsno7, prsno8,                                  *
*                   src, sfdte, sfcity,                                       *
*                   sfcty, sfst, intfd, sedte*
*-----*
* ACTIVE DATABASE FILES:  SOURCE.DBF                                       *
*
* ACTIVE INDEX FILES:    SRC.NDX, SPERS.NDX,                                 *
*                       PERSTYPE.NDX                                         *
*-----*
*****

```

```

use SOURCE.DBF index SRC,PERSTYPE,SPERS
append blank
replace SRC_NO with srcno
do case
  case cnt = 1
    replace PERS_NO with prsno
  case cnt = 2
    replace PERS_NO with prsno2
  case cnt = 3
    replace PERS_NO with prsno3
  case cnt = 4
    replace PERS_NO with prsno4
  case cnt = 5
    replace PERS_NO with prsno5
  case cnt = 6
    replace PERS_NO with prsno6
  case cnt = 7
    replace PERS_NO with prsno7
  case cnt = 8
    replace PERS_NO with prsno8
endcase

```

```
replace SRC_TYPE with src
replace DATE_SR_FD with sfdte
replace CITY_SR_FD with sfcity
replace CNTY_SR_FD with sfcnty
replace ST_SR_FD with sfst
replace INT_FD with intfd
replace DATE_SR_ET with sedte
use
```

```

*****
*                                     *
*                               ADDR*
*-----*
*
*   INVOKED BY:  BTHCERT.PRG, MRGCERT.PRG, *
*               DTHCERT.PRG                *
*
*   LOGIC:      THIS PROGRAM PROVIDES THE  *
*               ACTUAL ADDITION OF RELATION*
*               SHIPS INTO THE RELATION DA*
*               TASE.                       *
*-----*
*
*   PROGRAMS INVOKED:  NONE                *
*
*   GLOBAL VARIABLES:  srcno, prsno, prsno2, *
*                     prsno3, prsno4, prsno5, *
*                     prsno6, prsno7, prsno8, *
*                     rel                   *
*-----*
*
*   ACTIVE DATABASE FILES:  RELATION.DBF   *
*
*   ACTIVE INDEX FILES:    RSRC.NDX       *
*
*****

```

```

use RELATION.DBF index RSRC.NDX
append blank
replace SRC_NO with srcno
replace PERS_NO with prsno
do case
  case cnt = 1
    replace REL_PRS_NO with prsno
  case cnt = 2
    replace REL_PRS_NO with prsno2
  case cnt = 3
    replace REL_PRS_NO with prsno3
  case cnt = 4
    replace REL_PRS_NO with prsno4
  case cnt = 5
    replace REL_PRS_NO with prsno5
  case cnt = 6
    replace REL_PRS_NO with prsno6
  case cnt = 7
    replace REL_PRS_NO with prsno7
  case cnt = 8
    replace REL_PRS_NO with prsno8
endcase
replace RELATION with rel
use

```

```

*****
*                                     *
*                                     *
*-----*
*                                     *
*      INVOKED BY:  BTHCERT.PRG      *
*                                     *
*      LOGIC:       THIS PROGRAM PROVIDES THE *
*                   ACTUAL ADDITION OF BIRTH INFO *
*                   INTO THE BIRTH DATABASE.      *
*-----*
*                                     *
*      PROGRAMS INVOKED:  NONE          *
*                                     *
*      GLOBAL VARIABLES:  srcno, prsno, prsno2, *
*                   prsno3, prsno4, prsno5, *
*                   prsno6, prsno7, prsno8, *
*                   dtc, city, cnty, st, *
*                   fac, race, sex          *
*-----*
*                                     *
*      ACTIVE DATABASE FILES:  BIRTH_IN.DBF *
*                                     *
*      ACTIVE INDEX FILES:    BSRC.NDX, *
*                   SRCPERS.NDX *
*-----*

```

```

use BIRTH_IN.DBF index BSRC,SRCPERS
append blank
replace SRC_NO with srcno
do case
  case cnt = 1
    replace PERS_NO with prsno
  case cnt = 2
    replace PERS_NO with prsno2
  case cnt = 3
    replace PERS_NO with prsno3
  case cnt = 4
    replace PERS_NO with prsno4
  case cnt = 5
    replace PERS_NO with prsno5
  case cnt = 6
    replace PERS_NO with prsno6
  case cnt = 7
    replace PERS_NO with prsno7
  case cnt = 8
    replace PERS_NO with prsno8
endcase

```

```
replace BIRTH_DATE with dte
replace BIRTH_CITY with city
replace BIRTH_CNTY with cnty
replace BIRTH_ST with st
replace BIRTH_FAC with fac
replace BIRTH_RACE with race
replace BIRTH_SEX with sex
use
```

```

*****
*                               ADDMC                               *
*-----*
* INVOKED BY:  MRGCERT.PRG                                           *
* LOGIC:       THIS PROGRAM PROVIDES THE                             *
*              ACTUAL ADDITION OF MARRIAGE                           *
*              INFO INTO THE MARRIAGE DATABASE*                       *
*-----*
* PROGRAMS INVOKED:  NONE                                           *
* GLOBAL VARIABLES:  srcno, prsno, prsno2,                            *
*                   prsno3, prsno4, prsno5,                            *
*                   prsno6, prsno7, prsno8,                            *
*                   dte, city, cnty, st, fac,*                          *
*                   race, sex, age                                     *
*-----*
* ACTIVE DATABASE FILES:  MARRIAGE.DBF                               *
* ACTIVE INDEX FILES:    MSRC.NDX,                                    *
*                       MSRCPERS.NDX                                 *
*-----*

```

```

use MARRIAGE.DBF index MSRC,MSRCPERS
append blank
replace SRC_NO with srcno
do case
  case cnt = 1
    replace PERS_NO with prsno
  case cnt = 2
    replace PERS_NO with prsno2
  case cnt = 3
    replace PERS_NO with prsno3
  case cnt = 4
    replace PERS_NO with prsno4
  case cnt = 5
    replace PERS_NO with prsno5
  case cnt = 6
    replace PERS_NO with prsno6
  case cnt = 7
    replace PERS_NO with prsno7
  case cnt = 8
    replace PERS_NO with prsno8
endcase

```

```
replace MARRY_DATE with dtc
replace MARRY_CITY with city
replace MARRY_CNTY with cnty
replace MARRY_ST with st
replace MARRY_FAC with fac
replace MARRY_RACE with race
replace MARRY_SEX with sex
replace MARRY_AGE with age
use
```



```

*****
*                               ADDDC                               *
*-----*
*                               *
*   INVOKED BY:  DTHCERT.PRG   *
*                               *
*   LOGIC:       THIS PROGRAM PROVIDES THE *
*                 ACTUAL ADDITION OF DEATH INFO *
*                 INTO THE DEATH DATABASE.   *
*-----*
*                               *
*   PROGRAMS INVOKED:  NONE   *
*                               *
*   GLOBAL VARIABLES:  srcno, prsno, prsno2, *
*                       prsno3, prsno4, prsno5, *
*                       prsno6, prsno7, prsno8, *
*                       dte, city, cnty, st, *
*                       caus, race, sex, bdate, *
*                       bcity, bcnty, bst, fac *
*-----*
*                               *
*   ACTIVE DATABASE FILES:  DEATH_IN.DBF   *
*                               *
*   ACTIVE INDEX FILES:    DSRC.NDX       *
*                           DSRCPERS.NDX  *
*                               *
*****

```

```

use DEATH_IN.DBF index DSRC,DSRCPERS
append blank
replace SRC_NO with srcno
do case
  case cnt = 1
    replace PERS_NO with prsno
  case cnt = 2
    replace PERS_NO with prsno2
  case cnt = 3
    replace PERS_NO with prsno3
  case cnt = 4
    replace PERS_NO with prsno4
  case cnt = 5
    replace PERS_NO with prsno5
  case cnt = 6
    replace PERS_NO with prsno6
  case cnt = 7
    replace PERS_NO with prsno7
  case cnt = 8
    replace PERS_NO with prsno8
endcase

```

```
replace DEATH_DATE with dte
replace DEATH_CITY with city
replace DEATH_CNTY with cnty
replace DEATH_ST with st
replace DEATH_CAUS with caus
replace DEATH_RACE with race
replace DEATH_SEX with sex
replace BURY_DATE with bdate
replace BURY_CITY with bcity
replace BURY_CNTY with bcnty
replace BURY_ST with bst
replace BURY_FAC with fac
use
```

```

*****
*                               CRTSRCNO                               *
*-----*
* INVOKED BY:  BTHCERT.PRG, MRGCERT.PRG,                            *
*              DTHCERT.PRG                                           *
* LOGIC:      THIS PROGRAM CREATES A SOURCE                          *
*              IDENTIFICATION NUMBER FOR EACH NEW*                   *
*              SOURCE RECORD ENTERED AND PROVIDES*                   *
*              THE NUMBER BACK TO THE INVOKING *                       *
*              PROGRAM.                                              *
*-----*
* PROGRAMS INVOKED:  NONE                                           *
* GLOBAL VARIABLES:  srcno                                          *
*-----*
* ACTIVE DATABASE FILES:  SRCNO.DBF                                  *
* ACTIVE INDEX FILES:    NONE                                        *
*-----*
*****

*
* SINCE THE IDENTIFICATION NUMBER CREATED IS *
* EVENTUALLY STORED IN THE SRC_NO FIELD OF THE *
* DATABASE FILES, IT IS REQUIRED THAT THE RESULT *
* OF THIS NUMBER CREATION BE OF CHARACTER TYPE. *
* THE str FUNCTION CONVERTS THE NUMERIC VALUE TO *
* CHARACTER, WHILE THE val FUNCTION CONVERTS THE *
* CHARACTER VALUE TO NUMERIC SO THAT ARITHMETIC *
* CAN BE PERFORMED. *
*

```

```

use SRCNO.DBF
locate for NO = "0"
srcno = str(val(KEY)/10,6)
replace NO with srcno
use

```

```

*****
*                                     CRTPRSNO                                     *
*-----*
*
*   INVOKED BY:  BTHCERT.PRG, MRGCERT.PRG,
*                DTHCERT.PRG
*
*   LOGIC:       THIS PROGRAM CREATES A PERSON
*                IDENTIFICATION NUMBER FOR A NEW
*                PERSON AND PROVIDES THE NUMBER
*                BACK TO THE INVOKING PROGRAM
*-----*
*
*   PROGRAMS INVOKED:  NONE
*
*   GLOBAL VARIABLES:  cnt, prsno, prsno2, prsno3,
*                      prsno4, prsno5, prsno6,
*                      prsno7, prsno8
*-----*
*
*   ACTIVE DATABASE FILES:  PRSNO.DBF
*
*   ACTIVE INDEX FILES:    NONE
*-----*
*****
*
*   SINCE THE IDENTIFICATION NUMBER CREATED IS
*   EVENTUALLY STORED IN THE PERS_NO FIELD OF THE
*   DATABASE FILES, IT IS REQUIRED THAT THE RESULT
*   OF THIS NUMBER CREATION BE OF CHARACTER TYPE.
*   THE str FUNCTION CONVERTS THE NUMERIC VALUE TO
*   CHARACTER, WHILE THE val FUNCTION CONVERTS THE
*   CHARACTER VALUE TO NUMERIC SO THAT ARITHMETIC
*   CAN BE PERFORMED.
*
*

```

```

use PRSNO.DBF
locate for NO = "0"
if cnt > 1
  do case
    case cnt = 2
      prsno2 = str(val(KEY)/10,6)
      replace NO with prsno2
    case cnt = 3
      prsno3 = str(val(KEY)/10,6)
      replace NO with prsno3

```

```
case cnt = 4
  prsno4 = str(val(KEY)/10,6)
  replace NO with prsno4
case cnt = 5
  prsno5 = str(val(KEY)/10,6)
  replace NO with prsno5
case cnt = 6
  prsno6 = str(val(KEY)/10,6)
  replace NO with prsno6
case cnt = 7
  prsno7 = str(val(KEY)/10,6)
  replace NO with prsno7
case cnt = 8
  prsno8 = str(val(KEY)/10,6)
  replace NO with prsno8
endcase
else
  prsno = str(val(KEY)/10,6)
  replace NO with prsno
endif
use
```

INCORPORATING EXPERT SYSTEM TECHNOLOGY
INTO A PROFESSIONAL GENEALOGICAL
INFORMATION SYSTEM

by

BARBARA STOUT PARKER

B.S., Virginia Tech University, 1978

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Genealogy involves the comprehensive compilation and research of information on a group of people related by blood or marriage. The genealogist researches a vast amount of data in order to unravel the family lineage. Decisions based upon the data must be made, requiring a great deal of time and expert knowledge. This effort would be greatly enhanced by a database system that incorporates expert system technology. The expert system would facilitate the storage, linkage and manipulation of the data, and also assisting in the genealogical analysis and decision process.

This thesis documents a project whose objective was to design and implement a genealogical database system incorporating expert system technology. The thesis includes a discussion of the genealogical process, the expert system's requirements, design and implementation.

The features of the expert system include a base of rules provided by genealogical experts and the capability to add, modify, delete, or temporarily override these rules. It provides the ability to produce an audit trail of rules used to make a decision. It also includes an inference function that analyzes the data in the database, makes decisions, draws conclusions where possible, reports errors and warnings, suggests future strategies. A user-friendly interface is provided.