

DESIGN OF A PROFESSIONAL
GENEALOGICAL INFORMATION SYSTEM
INCLUDING NAVIGATION FROM AN
UNSTRUCTURED DATABASE TO A
STRUCTURED DATABASE

by

ELLEN J. BAILEY

B.S., East Tennessee State University, 1980

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:

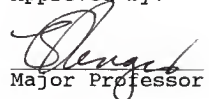

Major Professor

Table of Contents

Chapter 1 - Overview.....	1
1.1 Introduction.....	1
1.2 User Community.....	2
1.3 Literature Review - Overview.....	6
1.4 Genealogy Literature.....	7
1.5 Database Design Literature - Overview.....	8
1.6 Database Design Methodology - Unger/Fisher.....	9
1.6.1 Predesign Evaluation.....	9
1.6.2 Information Modeling.....	10
1.6.3 Semantic Modeling.....	11
1.6.4 Logical Database Design.....	12
1.6.5 Data-Base Management System Selection.....	13
1.6.6 Cost Benifit Analysis.....	13
1.6.7 Physical Design and Implementation.....	14
1.7 Classical Database Design Models.....	14
1.7.1 Hierarchical Model.....	14
1.7.2 Network Model.....	15
1.7.3 Relational Model.....	16
1.8 Null Value Literature.....	19
1.9 Text Manipulation Literature.....	21
1.10 Rule_Based Expert Systems Literature.....	23

LD
0668
.T4
CRNSC
1988
B34
C.2

1.11 Summary of the Professional Genealogical Information System Project As It Relates to the Literature.....	26
Chapter 2 - Genealogical Processes.....	28
2.1 Introduction.....	28
2.2 Amateurs.....	28
2.3 Professionals.....	30
2.4 Summary.....	36
Chapter 3 - Requirements.....	37
3.1 Introduction.....	37
3.2 General Overview.....	38
3.3 Specific Details.....	40
3.3.1 Data-Base Control Subsystem.....	42
3.3.2 Text Manipulation Subsystem.....	43
Chapter 4 - Design.....	44
4.1 Introduction.....	44
4.2 General Overview.....	46
4.3 Data-Base Control Subsystem.....	49
4.3.1 Genealogy Database Design.....	51
4.3.2 Data-Base Front End.....	64
4.3.3 Data-Base Management System (DBMS).....	77
4.4 Text Manipulation Subsystem.....	78
4.4.1 Family Chronicle Model.....	81
4.4.2 Text Management Syntax.....	83
4.4.3 Text Management System - Text Database....	85

4.5	Summary.....	90
	Chapter 5 - Implementation.....	91
5.1	Introduction.....	91
5.2	Genealogy Database Structure.....	92
5.3	Data-Base Management System - PIGS.....	95
5.3.1	User Interface.....	96
5.3.2	Add Function.....	96
5.3.3	Delete Function.....	98
5.3.4	Update Function.....	99
5.3.5	Query/Report Function.....	100
5.4	Summary.....	103
	Chapter 6 - Conclusions and Extensions.....	105
6.1	Introduction.....	105
6.2	Conclusions and Extensions for Requirements and Design.....	105
6.3	Conclusions and Extensions for the Implementation.....	108
6.3.1	Activation of Multiple Index Files.....	109
6.3.2	Relating Multiple Database Files.....	109
6.3.3	Non-Character Key Attributes.....	110
6.3.4	Summary of Conclusions and Extensions for the Implementation.....	112
6.4	Summary.....	113
	References.....	114
	Appendix A: Bernstein Algorithm.....	116

Appendix B: Pedigree and Family Unit Charts.....	117
Appendix C Implementation Code.....	121

LIST OF FIGURES

Figure 3_1.	Professional Genealogical Information System	41
Figure 4_1.	Professional Genealogical Information System: Expanded View.....	45
Figure 4_2.	Data-Base Control Subsystem.....	50
Figure 4_3.	Entity-Relationship Diagram.....	52
Figure 4_4.	E-R Source Entity.....	53
Figure 4_5.	E-R Person Entity.....	53
Figure 4-6.	E-R Birth and Death Relationships.....	55
Figure 4_7.	E-R Marriage and Military Relationships..	56
Figure 4_8.	E-R Religion and Residence Relationships..	57
Figure 4_9.	E-R Occupation and Land Relationships....	58
Figure 4_10.	E-R Tax and Relation Relationships.....	59
Figure 4_11.	E-R Qualifying Relationship.....	60
Figure 4_12.	Text Manipulation Subsystem Components...	79
Figure 4_13.	Family Chronicle Model.....	82
Figure 4_14.	Text Management Syntax.....	84

LIST OF TABLES

Table 4_1	Relations.....	62
Table 4_2	DBFE Pseudo-Code Add Description.....	67
Table 4_3	DBFE Pseudo-Code Delete Description.....	74
Table 4_4	DBFE Pseudo-Code Update Description.....	75
Table 4_5	DBFE Pseudo-Code Query Description.....	76
Table 5_1	Database File Structure.....	92
Table 5_2	Index File Structure.....	94

CHAPTER 1 - OVERVIEW

1.1 Introduction

This thesis addresses the design of a professional genealogical information system. This paper presents the requirements, design and implementation of such a system. Genealogy was chosen as the subject for this research since it readily presents itself with problems that are of interest in the computer industry.

Upon initial review of the prevalent genealogical processes it was apparent that a computerized system would require an extensive database design effort involving:

- 1) the incorporation of theories associated with null values,
- 2) a technique for providing user friendly search, comparison and extrapolation capabilities,
- 3) a method for navigating from text to a database
- 4) the incorporation of technologies associated with rule-based expert systems.

This paper addresses areas one, two and three in detail. Discussion about rule-based expert system technology will be limited to the degree that it impacts the database

design effort. However, the rule-based expert system portion of the overall genealogical system is being addressed in another Master's thesis at Kansas State University (BSP88).

The remainder of this chapter presents a discussion on the user community and a review of the current literature. The literature review discusses Database Design, Null Values, Text Manipulation and Expert Systems to the degree that they apply to the genealogical information system. Additional genealogical literature is also reviewed for the purpose of providing background information and because there is no literature currently available that addresses the areas specified above in direct relation to genealogy.

The remainder of the paper will address prevalent genealogical processes (Chapter 2), the requirements of the system (Chapter 3), the design of the system (Chapter 4), the implementation of the system (Chapter 5) and conclusions and extensions (Chapter 6).

1.2 User Community

At this point it is important to emphasize that this system design effort is directed toward the development of a tool that can be used not only by amateur genealogists

but also by professional genealogists. The overall tasks and needs of the two groups differ tremendously. This section is devoted to explaining those differences and emphasizing why the professional system is more challenging.

The obvious distinction between the two groups of genealogists is that one receives monetary compensation for their efforts while the other doesn't. From this perspective, one could assume that amateurs could simply extend their experience into a money-making venture if they so choose. This assumption is valid and introduces another important distinguishing issue. There are professional genealogists and there are certified professional genealogists. There are several certification levels associated with genealogy and each requires extensive testing before the certification is granted. The certification levels are as follows:

- 1) Certified Genealogical Researcher - this certification requires the genealogist to exhibit the capability to find, interpret and report records.
- 2) Certified American Lineage Specialist - this certification requires the genealogist to exhibit the capability to document family lineages acceptable to lineage societies.

- 3) Certified American Indian Lineage Specialist-
this certification is identical to # 2 except
that it requires the genealogist to have an
extensive knowledge of the American Indian.
- 4) Certified Genealogist - this certification
requires the genealogist to exhibit the
capability to research, document and prove
family genealogies.
- 5) Certified Genealogical Lecturer - this certi-
fication is an extension of # 4 but requires
The genealogist to exhibit lecturing
capabilities in all areas of genealogy.
- 6) Certified Genealogical Instructor - this
certification is also an extension of # 4 but
requires the genealogist to exhibit
instructional capabilities in all areas of
genealogy.

In this light, it is important to clarify that the genealogical database system is actually directed toward the amateur, professional and certified professional genealogical communities. For the purpose of simplification the term professional genealogist will be used to mean certified professional genealogist.

Other distinguishing characteristics between the amateurs

and professionals include the amount and type of research performed, the accuracy level of the research and the final results produced. The amateur, in most cases, does not perform as thorough an investigation as does the professional. In some cases, this could be due to lack of knowledge, i.e., not knowing what types of records, beyond the obvious, are actually available or not knowing the laws that impact the records involved. The professional genealogist must have or obtain an understanding of the state, county and local laws as well as where legal records are kept and how they pertain to the research at hand. Additional legal knowledge required by the professional genealogist consist of knowing the pertinent laws associated with the time period and place of an individual being investigated. For example, knowing that in 1807 in Guilford County, North Carolina a man could not obtain a land deed until he was at least 18 years of age could become a significant fact when trying to determine the age of an individual. In other cases, when documented proof of accurate research is not required and/or available records provide sufficient information, amateurs may not perform as thorough an investigation. The professional genealogist must at all times have documented proof of the results obtained or must denote that some doubt still exists and explain the reason for that doubt.

The most distinguishing characteristic between the two groups, is probably the final results of the genealogist's effort. Amateurs are normally concerned only with their families; therefore, the production of pedigree, descendant or family charts is sufficient as a final output. These type of outputs provide brief vital facts about each ancestor or descendant discovered but do not provide any additional detailed information. The final results of the professional genealogist's effort, however, is a compiled chronicle of a family's history. This type of output not only documents the vital facts but also provides information regarding the economic, social and religious environments at the time of each ancestor's or descendant's life. This is important because knowing more about the history of the surroundings provides a better understanding of the importance of each fact obtained.

1.3 Literature Review - Overview

Most of the current literature on Genealogical Computer Systems is written from either a genealogy perspective or a computer systems perspective, but to this date it has been impossible to find literature written by an author knowledgeable in both areas. Therefore, the following literature review is divided into five sections

concentrating on the research performed in the areas of Genealogy, Database Design, Null Values, Text/Data Manipulation and Rule-Based Expert Systems.

1.4 Genealogy Literature

The predominant request of the computer industry by genealogists is to provide a system that: performs all the required genealogical functions; makes it simple to use for the computer novice; can be used on a personal computer at home and is reasonably priced (LA86).

Now that personal computers are more affordable, genealogists are eager to be saved from the reams of pulp under which they are currently suffocating. Therefore, the problem becomes finding the software that meets the needs specified above. Currently the types of products being utilized by genealogists include: 1) Data-Base Management Systems (DBMS), 2) word processing systems and 3) specialized genealogical packages. Each of these product types have features that are better suited for certain functions (RP83). Since the genealogical process involves researching information; collecting descriptive information, in the format of brief facts or text; storing information; linking all associated information; manipulating information and retrieving information to produce desired outputs (HFML86), it is easy to understand

why any one of the three product types listed above appear appropriate to the user. However, each one of the product types presents problems for the genealogist. The DBMS and WP systems require some computer skills in order for them to be truly useful tools; therefore, those genealogists that have an overwhelming fear of computers shy away from these tools. The available tailored genealogical packages, though sometimes easy to use, tend to limit the flexibility of the user due to their preformatted design. The biggest problem of all, however, is that there isn't a tool that combines the capabilities of all three of the software types mentioned above.

1.5 Database Design Literature - Overview

In the development of computer software systems today, in particular a database system, there are many criteria that must be addressed before proceeding with development activities. It is necessary to reduce data redundancy, provide stable data structures that can be readily changed upon users' requirements changing, allow users the capability of making ad hoc requests for data, maintain complex relationships between data elements, and support a wide variety of decision needs (ITH84). To effectively address these criteria requires proper design techniques and procedures. The techniques are tools used by the

database designer while the procedures are rules for determining the sequence in which the techniques are applied. The combination of the techniques and procedures used is often referred to as a design methodology. The design methodology chosen for design of the professional genealogical database system is the Unger/Fisher Methodology. Section 1.6 will describe the methodology and section 1.7 will present an overview of the three classical, logical database models.

1.6 Database Design Methodology - Unger/Fisher

The Unger/Fisher Database Design Methodology was created by Dr. Elizabeth A. Unger and Dr. Paul Fisher, professors at Kansas State University. This section will present the seven phases of their methodology.

1.6.1 Predesign Evaluation

The predesign evaluation phase involves an organizational survey, assessment of the functions (in terms of data support) and assessment of the potential for a database system. The organizational survey involves determination of:

- 1) the functions performed by the user organization,
- 2) the types of forms currently processed,
- 3) the types of problems that exist in the organization's current environment.

The assessment of the functions involves a preliminary analysis of the functions performed. The functions must first be identified. Details regarding the inputs, outputs and data needed must then be obtained for each function. A functional assessment should also determine the volume and frequency of use, availability, security and integrity characteristics. After completing the organizational survey and the functional assessment, the potential for a database system can be assessed.

The basic tool used in this phase is interviews with users in the organization. The products of the predesign evaluation should include a preliminary functional analysis (which may change over time), a compendium of forms used and processed and a documented evaluation of potential for effective database usage.

1.6.2 Information Modeling

The information modeling phase involves data collection and analysis, requirements analysis, relationship analysis

and entity analysis. Data collection and analysis involves defining the data elements and collecting the associated records, files and forms. Requirements analysis involves defining the users' views and identifying the outputs, the users of the outputs and the functions of the outputs. The relationship analysis determines the data attributes and defines dependencies. At this point the designer should also determine how data are identified, filed and retrieved. In the entity analysis, dependencies are normalized to form entities and the relationship between the entities is derived (one to one, one to many, many to many).

The tools normally used in the information modeling phase include Requirement Analysis Methodologies and an automated Data Dictionary System. The products of this phase then become a data dictionary, a formal definition of relationships and a functional requirement specification.

1.6.3 Semantic Modeling

The semantic modeling phase puts emphasis on the object level rather than the data element level of information. This phase involves the creation of a semantic data model (SDM) or conceptual graph model. A SDM provides, in a programming language type syntax, a meaning and semantic

description of the entities in a database. A conceptual graph displays the same type of information only in graphical form.

The tools used during the semantic modeling phase are popular SDM Methodologies or Conceptual Graph Techniques and in particular those developed by Hammer and McLeod or Sowa, respectively. The obvious product of this phase is an accurate SDM or conceptual graph.

1.6.4 Logical Database Design

At this point in the design life cycle, the designer should have tools available that the user can understand. This phase involves using the entities and relationships developed in the previous phases and translating them into a logical database model. As of this date, the translation process is entirely manual. There are three classical, logical database models which are discussed in more detail in section 1.7.

The tools used in the logical database design phase normally include SDM's or conceptual graphs and Enterprise Methodologies. The products of this step should be a conceptual representation of the database in the form of an Entity/Relationship (E/R) diagram and model schema.

1.6.5 Data Base Management System (DBMS) Selection

Based on the logical model chosen in the previous phase and the hardware that is going to be used, a selection of a DBMS must occur at this point. This step is often dependent on available products or organizational standards.

The major tools used during this step are the logical database model, available DBMS documentation and organizational standards. The product of this step should obviously be an appropriate DBMS.

1.6.6 Cost Benefit Analysis

At this stage of the design life cycle it is appropriate to determine the cost benefits of a database system. After performing the five previous steps, the designer should have a good understanding of the overall system requirements.

The tools used during this phase are less rigid than those of previous phases but could include things such as a cash flow and analysis program. The product of this phase should be a documented final analysis including detailed cost information.

1.6.7 Physical Design and Implementation

The final phase of the design process involves translating the logical design to a DBMS model and external schema, designing the storage and access methods, coding the schema and subschema, testing the product and implementing the product.

The tools used in the physical design and implementation phase could include popular Design Methodologies and appropriate programming tools. The obvious product of this final step is a tested and functional database system.

1.7 Classical Database Design Models

When considering an extensive database design, such as the professional genealogical system, it is appropriate to understand the three classical design models of logical databases. This section will describe those classical design models.

1.7.1 Hierarchical Model

The hierarchical model organizes data in simple tree structures and a collection of these structures represents a hierarchical database. A tree is composed of a hierarchy of elements (object classes) referred to as

nodes. The uppermost level of the hierarchy has one node which is called the root. Each node (or object class) is restricted to having only one parent node but each parent node can have multiple child nodes. This type of model obviously requires that some nodes be subordinate to others.

Hierarchical databases are normally presented with structure diagrams representing the nodes as rectangular boxes and the parent-child relationships as one directional links from parent to child. Hierarchical databases are usually implemented, or data is accessible, by either tree traversal or general selection. In a hierarchical database a given node can only take on its full meaning when viewed in the context of the entire hierarchy (JDU82, LB&JPG86, ITH84, CJD86, JM76).

1.7.2 Network Model

The network model was proposed by the Data Base Task Group (DBTG) of the Conference On Data Systems Languages (CODASYL) in 1971. This model was the first attempt at implementing a database standard.

The network model can be regarded as an extension of the hierarchical model with the major difference being that the network model allows a child object class to have zero

or more parents. The network model defines a set concept often referred to as owner-coupled set. This concept defines a set as a group of objects each of which consists of one owner (parent) object and zero or more member (child) objects. In the network model, like the hierarchical model, a given object class can only take on its full meaning when viewed in the context of the entire network (JDU82, LB&JPG86, ITH84, CDJ86, JM76).

1.7.3 Relational Model

The relational model was first proposed by Dr. E. F. Codd in a seminal paper in 1970. The relational model represents data in the simple form of a two dimensional table referred to as a relation. Each column of the table is referred to as an attribute and each attribute must have a unique name. The order of the attributes has no significance. The entries in any column are all of the same kind, i.e. an attribute has a defined set of values known as its domain. Different attributes can, however, have the same domain; this is why unique attribute names are so crucial. Each row of the table is known as a tuple. There can be no duplicate tuples. In a relation with n columns, each row is referred to as a n -tuple. Also, this relation is of degree n .

One of the most important aspects of the relational model

is that of a key. A key is the attribute or set of attributes that uniquely identifies tuples in a relation. There are basically two properties that a key should possess: 1) Uniqueness - the set of attributes takes on a unique value in the relation for each tuple, 2) Nonredundancy - if an attribute is removed from the set of attributes, the remaining attributes do not possess the uniqueness property (ITH84). A prime attribute is then defined as an attribute that is part of at least one relational key and a nonprime attribute is not part of any key.

To represent user information by relations that do not create anomalies following tuple operations (add, delete, update), the relation must be in normal form. Normalization is a process of altering structures of relations so that new relations have desirable properties for particular manipulations. There are six widely accepted normal forms: First, second, third, fourth and fifth normal forms (1NF, 2NF, 3NF, 4NF and 5NF, respectively) and Boyce Codd Normal Form (BCNF). Normalization is related to the logical description of the data and not to the physical description. This allows the user's views of data to be kept independent from the physical representation of data; therefore, the physical representation and the hardware can be changed without

changing the logical or user's view.

To completely understand normalization it is important to first understand data dependency. Data dependency is a semantic constraint which describes the possible tuples which can occur in a relation. There are approximately 15 types of dependencies but the three most common ones are functional (FD), multivalued (MVD) and join (JD). Functional dependency is the most appropriate to discuss at this point. An attribute set X is said to be functionally dependent on an attribute set Y if when given a value for every attribute in Y , a unique value for the set of attributes in X is determined. This is represented as $Y \rightarrow X$ and is read as Y functionally determines X . Functional dependencies must be derived by the database designer and are dependent on real world situations. The set of FDs that describes a database completely is called a cover and more FDs can be derived from the cover using Armstrong's Axioms (Reflexivity, Augmentation, Transitivity, Union, Pseudotransitivity and Decomposition). The closure is then known as the minimum set of dependencies that can be derived from Armstrong's Axioms.

With most of the discussion on relational terminology completed, it is appropriate to discuss normalization

methods. There are two basic normalization methods:

- 1) Synthesis - start with functional dependencies and derive relational database,
- 2) Decomposition - start with relational database and derive a better relational database.

The decomposition algorithms are normally complex; therefore, the synthesis algorithms are more popular. Perhaps the best known synthesis algorithm is the one derived by P. A. Bernstein (PAB76). Appendix A contains the six steps defined in Bernstein's final algorithm (Bern 2). Bern 2 is automated, always derives 3NF and always provides the minimal number of relations. There are two important assumptions associated with Bernstein's algorithm:

- 1) All connections among attributes in a database description can be represented by FDs.
- 2) Any given FD has but one meaning; the assumption behind this is that for any two sets of attributes X and Y there is at most one FD, $X \rightarrow Y$ (Uniqueness Assumption).

1.8 Null Value Literature

Since genealogists are constantly confronted with unknown

or incomplete data, null values need to be addressed. There are two types of null values that are relevant to the genealogical problem: no currently known value and no value is valid for this attribute.

Information published by Ms. Maria Wilson presents an approach for the inclusion of null values in relational databases (MW85). Wilson defines a representation of two distinct null values that exhibit desired processing and behavior qualities. A description of modified truth tables that ensure the integrity of results when nulls are allowed in the evaluation of conditions is also presented. Logical and algebraic operators are defined for data extraction and manipulation capabilities. Wilson's approach relaxes functional dependency constraints on nulls by defining restrictions which allow flexibility and at the same time prevent semantic contradictions.

The inclusion of null values in a database, in particular a relational database, introduces three types of integrity problems. The first problem involves keys. If null values are allowed as keys, distinguishing between tuples or records of the database is a problem. The next problem involves the evaluation of conditions. If null values are allowed in the evaluation of conditions, ambiguous results could occur. Desired records may or may not be included.

The third problem involves relational join operations. If null values are allowed in join operations, surprising results can occur. Desired tuples may be excluded, while extra tuples with incorrect semantics could result.

1.9 Text Manipulation Literature

In a genealogical system, the ability to incorporate data values from the structured genealogical database into an unstructured text database would be a useful asset. Professional genealogists' basic form of output is a family history book which consists of the vital facts about an individual embedded in text. Some genealogists use word processing systems to create this text output. To use a word processing system, however, the genealogist must have ready access to all the pertinent data about each individual discussed in order to embed it into the document. If the genealogist could produce such a document and automatically draw the pertinent data from an existing genealogy database, the procedure for creating the final document would be less cumbersome. It would not require manually searching through files of data collected to compile the final document.

This concept lead to research in the area of personal computer products that supposedly integrate database and

text manipulation capabilities. One product that stood out above others was Lotus 1-2-3. The Lotus 1-2-3 product initially claimed to have triple functionality (spreadsheet, database and graphics), hence its name 1-2-3. Later Lotus announced its new Lotus Report Writer software for text manipulation capabilities.

It is important to explain at this point that the Lotus definition of database only incorporates the ability to sort, search and link data within one spreadsheet (table). The important fact to understand here, however, is that Lotus 1-2-3 does not provide the capability to relate multiple spreadsheets (tables) or extract information based on a variety of criteria. It may be possible to write macros to link tables in Lotus but even Alexander Crosett, III, a marketing manager at Lotus Development admits, "It is not a simple operation" (MA86). With this thought in mind, review of Lotus' Report Writer was undertaken. The Report Writer package basically allows a user to specify a spreadsheet and print out all or some of the fields at a specific point within a document (GM86). Since Lotus 1-2-3 was actually introduced for its financial spreadsheet capabilities, Report Writer becomes more a tool for producing management reports with condensed versions of the spreadsheets included. Although this is a powerful and useful tool in the financial

community, it does not quiet address the needs of genealogists. The genealogists need to embed particular facts into their sentence structures instead of just providing facts as a table or chart within a document.

1.10 Rule-Based Expert Systems Literature

In the field of genealogy, there is a massive amount of research that must be undertaken which correspondingly produces a massive amount of data. Once the data is available, genealogists must constantly query the data and evaluate the facts associated with an investigation. The techniques and degrees of performing research and evaluation are varied due to individually differing approaches (e.g., that of amateurs and professionals). There are, however, some basic rules of thumb that are used when performing evaluations and analyses. Even rules as simple as John Adams Sr. cannot be the father of John Adams Jr unless he is older are pertinent. If a computerized system could help the genealogist analyze the data and realize information of this sort a lot of time and effort could be saved. In addition, there are also rules that are applicable to the problem of unknown or incomplete data (null values). The realization that basic rules are used throughout the genealogical process introduces the need for integration of a rule-based expert

system into the overall professional genealogical database system. When developing the database design for the genealogical system, this integration must be considered. Since the rule-based expert system portion of the genealogical system is beyond the scope of this paper, the literature search presented below only addresses the extent to which it effects the database design effort.

A rule-based expert system involves acquiring the knowledge needed and structuring it into a usable form. To obtain the knowledge, requires interaction with experts of the field. There are basically two approaches being researched today for the integration of expert system capabilities into a database system. Before discussing these, it is appropriate to present how the Artificial Intelligence (AI) community is differentiating between a database and a knowledge base. A database is a collection of facts (data). The amount of data is usually large, it can change over time and updates can be made routinely. The correctness of the data can be objectively determined by comparing the values with real-world observations. A knowledge base, on the other hand, contains information at a higher level of abstraction. Knowledge bases become feasible in areas where problems are difficult to address by algorithmic methods. These systems describe classes of objects rather than individual objects. The objective of

a database is to effectively manage the data needed in the user environment, while the objective of a knowledge base is to produce information suitable for decision making processes (GW84).

The first approach commonly discussed for providing integration of database and knowledge base (in particular rule-based) technologies involves keeping the knowledge base entirely separate from the database. This allows the rule-based system to be designed at a higher level of abstraction. The rule-based system will then have an interface to the database in order to perform its decision making processes. The two most critical issues in this type of system approach become the query language interface and the knowledge flow. Most queries to databases are not phrased in a user language but instead they are executed by providing parameters to a transaction program that typically limits flexibility. In today's Data Base Management Systems (DBMS) realistic queries are difficult to achieve. The flow of knowledge on the other hand comes from the experts or the rule-base. The rules must be in machine processible form in order for the system to interact with the objects that the knowledge is about, or in other words, the database (GW84).

The second approach discussed in the literature involves

integrating the knowledge base directly into the database. In order to have an intelligent database, a more semantic view of the data is needed. In addition, the need for a natural language as a user interface becomes more prominent (LB&JPG86).

Studies along these lines are showing that there is great promise for future application of artificial intelligence techniques into database technology. As the development of new database models concentrates more and more on the semantics of the data, this type of integration becomes even more feasible (LB&JPG86).

1.11 Summary Of The Professional Genealogical Information System Project As It Relates To The Literature

The objectives of this project, as specified in the introduction, are to design and implement a Professional Genealogical Information System that incorporates:

- 1) theories associated with null values,
- 2) techniques for providing user friendly search, comparison and extrapolation capabilities,
- 3) methods for navigation from text to data,
- 4) technologies associated with rule-based expert systems.

The literature revealed that no genealogical systems

address the areas of concern for this project. The goal of the literature search then became determination of the most predominant technologies and theories in the areas of concern. The areas investigated were Genealogy, Database Design, Null Values, Text Manipulation, and Rule-Based Expert Systems, with extra emphasis on the database design literature.

In summary, this thesis establishes a genealogical information base and solves the problem of navigating from the unstructured text database to the structured genealogy database.

CHAPTER 2 - GENEALOGICAL PROCESSES

2.1 Introduction

Chapter one briefly introduced and explained some key differences between the amateur and professional genealogists. Some terms regarding data, sources and final outputs were mentioned. The purpose of this chapter is to expand on some of those terms in the hopes of creating an appreciation for the genealogical processes and the complexities associated with them. This chapter will divide the processes into two categories; one for the amateurs and one for the professionals.

2.2 Amateurs

Two important distinguishing characteristics between the amateurs and professionals are: 1) the amount and type of research performed and 2) the accuracy level of the research performed. Both of these characteristics involve the research process and the records used. Since these are areas of more significance to the professionals, they will be discussed in more detail in the next section. This section will, therefore, concentrate on a third distinguishing characteristic, the final results produced.

Amateurs are generally interested in only their families and being able to establish links as far back as possible.

This typically means that the production of pedigree, descendant or family unit charts is sufficient as output for their efforts. The pedigree chart, or line-of-ascent, begins with the most recent descendant (who is usually the amateur or person performing the research) and traces the ancestors as far back as can be found. The descendant chart is basically the same except that it begins with a forefather and builds up to the most recent descendants. Since both of these charts are similar except for the order of presentation, this discussion will concentrate on the pedigree chart. The ancestors represented on the pedigree chart are only the father-mother pairs (family units) and do not include children of those pairs. This is why a pedigree chart is usually followed by a family unit chart. A family unit chart is made for each male ancestor that appears on the pedigree chart. The family unit chart contains the names and facts of the father, mother, wife and children of the male ancestor in question. In order to associate pedigree charts with family unit charts and therefore establish family linkages, it is essential to use some type of numbering scheme for the charts and their association as well as the individuals and their linkages. The numbering scheme for the association of charts is usually fairly simplistic. Each pedigree chart is given a number (beginning at one)

and each family unit chart will denote the corresponding number based on the male ancestor being documented. The family unit charts, likewise, have numbers associated with them that are denoted at the appropriate place on the appropriate pedigree chart. The numbering schemes for individuals are somewhat more difficult. There are three widely acceptable numbering schemes for individuals: 1) Personal, 2) Family Unit and 3) Continuation (HMFL80). Some genealogists elect to use these number schemes while others choose to create their own. For this reason it is not important to understand details on the numbering schemes but instead to understand that this type of linking is not a simple task when several generations are spanned. Appendix B contains copies of two pedigree and two family unit charts with fictional data included. By reviewing these charts, the type and extent of data important to the amateurs should be apparent.

2.3 Professionals

Professional Genealogists also make use of pedigree, descendant and family unit charts. However, these charts are not the end of the effort, but instead only a means to the end. Professional Genealogists, like the amateurs, also have to have an orderly means of indexing or linking the data found. Therefore, the use of the charts usually

serves this purpose in the professional community. The actual end (or result) of their overall effort is a historical family chronicle or genealogy. There are actually some specific differences between a family chronicle and a genealogy (HMFL80), however, they are not of grave importance to this discussion. The important point to remember here is that a more detailed output is produced by the professional.

The professionals not only want to document the vital facts (names, ages, dates, etc.) but also the minute details regarding the economic, social and religious environments surrounding the facts. In order to produce detailed and accurate family chronicles, the professional must pay careful attention to the records researched. The remainder of this section will present some of the important types of records available, the information ascertained from the records and the complexities associated with the research of these records.

The first type of records is Vital Records. Vital records can include marriage certificates, birth certificates, death certificates, divorce proceedings, newspapers, Family Bibles, cemeteries, and word of mouth. These records are where amateurs spend a great deal of their research effort and is an important starting point for

professional genealogists. Some of the important data that can be ascertained from these records are names, birth dates, parents' names, dates of marriages, spouses' names, marriage witnesses, dates of deaths, ages at deaths, places of burials, previous spouses' names, etc. These data not only provide important facts but also provide leads to other records that can be researched. The genealogist normally starts with one individual and builds up to other individuals based on information discovered. For example, in this case an individual's parents' names can possibly be obtained from a birth certificate and then research into their vital records can follow. As vital records of various individuals are collected, the genealogist can denote discrepancies in dates, names, etc. and possibly gather clues for additional research. For example, the detection of a difference in an individual's given name and the name he goes by could indicate a legal name change and therefore require research based on the new name. If discrepancies are detected among the vital records of one individual, e.g. date of birth varying between a birth certificate and Family Bible, the genealogist must make a decision as to which record takes precedence. The type of activity described here, beyond the simple collection of facts, is where professionals tend to excel beyond the amateurs.

Another type of records is Census Records. The type of information that can be obtained from census records includes places of residence, dates of residence, people living in households, occupations, ages, educational data (literate or not), etc. From this data a genealogist could possibly detect an indication of a second marriage due to gaps in the ages of the husband and wife. Census records are important for determining migration patterns which are of significance to the professional genealogist because it aids in telling a family's history. Information such as grandparents or in-laws resident in a household also helps complete the story.

A third type of records is Wills. Wills can provide names of children, grandchildren, parents, etc. which are helpful for future research. Information regarding the economic status of individuals can be ascertained, another important "story telling" fact. Wills can be compared with marriage records for possible determination of ages of sons or unmarried daughters. Wills can provide information such as children being omitted from an estate. Wills can be compared to other wills for determining possible family traditions or chains of inheritance.

A fourth type of records is Estate Records. Estate Records can include bonds, inventories, bills, receipts,

estate sales, etc. These records can provide dates, economic lifestyle, occupation, educational level, names of widows, children, neighbors, business partners, etc. These records can be used in comparison with deeds to help the genealogist determine if one individual is actually the same as another individual discovered with the same name. For example, if an individual signed his name to all of his bills, he is probably not the same individual that signed a deed with an X.

A fifth type of records is Land Records. Land records can provide data regarding the locations of landmarks and adjoining properties and the owners. Land records in conjunction with laws could possibly be used in narrowing down the ages of individuals. For example, in Russell County, VA a man could buy or be granted land before the age of 21, but he could not sell land before the age of 21.

A sixth type of records is Tax Records. Tax records can provide names of slaves, ages of slaves, number of cattle owned, number of gold watches owned, etc. This type of data is helpful in ascertaining the socio-economic status of an individual. Another piece of information that can be uniquely obtained is that of adjoining land owners. Often times tax records were listed by person's name and

the order of the names (if not alphabetical) indicated adjoining land owners.

A seventh and final type of records is Court Records. Court records normally include lists of petitioners, jurymen, road workers, etc. When using these types of records, the professional genealogist will look at all names on the lists retrieved because they can provide hints toward business partners or neighbors. For example, if a man helped maintain a road it was normally the road he lived on; therefore, any other people listed as workers on that road could be neighbors.

The information presented in this chapter was provided by Ms. Helen Leary, a North Carolina Certified Professional Genealogist Instructor, via interviews and a book that she helped write and edit (HFML80).

It should be apparent at this point that genealogy is not only an art but also a science. This section has attempted to present the major types of records available, some of the data that can be obtained from the records and the complexities associated with the research of these records. Most of the information discussed beyond the vital records presents the efforts of the professional genealogist as opposed to the amateur.

2.4 Summary

The need for a computerized system to assist genealogists in their record collection, decision making processes and production of final results should be evident after understanding even the brief examples presented in this chapter. A database is needed to provide a data storage mechanism and to manage the linking of associated data. If techniques can be developed to provide user friendly search, compare and extrapolation capabilities, genealogists could at a minimum avoid the manual linking of associated data. A computerized system with built in intelligence could help genealogists in their decision making processes and also possibly provide information to amateurs that is currently beyond their scope of knowledge.

CHAPTER 3 - REQUIREMENTS

3.1 Introduction

The objective of this project is to design and implement a professional genealogical information system. According to genealogists, to have a useful system it must be able to :

- 1) provide easy to use store, search and extrapolation capabilities,
- 2) handle unknown or incomplete information,
- 3) store and manipulate text interchangeably with data and
- 4) assist in the decision making process.

In order to address these needs, it is helpful to think of the system as four separate subsystems:

- 1) the user front end subsystem which will address the ease of use requirement,
- 2) the data-base control subsystem which will address the data storage, query and extrapolation requirement,
- 3) the text manipulation subsystem which will address the text handling requirement and
- 4) the rule-based expert subsystem which will address the decision making requirement as well as the handling of unknown or incomplete information.

This chapter will present the requirements of the system with concentration on the data-base control and text manipulation subsystems. The user front end and rule-based expert subsystems are being addressed in another

Master's thesis at Kansas State University (BSP88). Section 3.2 will give a general overview of the requirements and section 3.3 will present more details on specific functions.

3.2 General Overview

The Professional Genealogical Information System must be suitable for use on a personal computer since this system will be used, for the most part, by genealogists at home or in small offices. The system must provide soft-fail capability and provide easy to use diagnostics. A Data Base Management System (DBMS) suitable for the personal computer will be required to manage the storing, linking, searching and extrapolation of data. An easy to use external interface will be required to allow genealogists (non-programming types) simple access to the system. The user front end subsystem will provide this capability.

The system must allow the user the capability to add, delete, update, query and report the data. The system must provide preformatted input screens for all standard record types used by genealogists. The genealogist must be able to select the record type needed from an online command menu. The system must allow the genealogist the capability to change the standard record formats, i.e. add or delete fields, to suit their needs. The system will

not, however, allow the online addition of new record types. All standard types of genealogical outputs (reports, charts, documents, etc.) must be available from the system.

The system must provide text manipulation capabilities and support navigation between text and data. The system must allow the user the capability to create free-form documents and it must also automatically provide data for insertion into documents.

The system must be able to establish a dialogue between the data-base control subsystem and the rule-based expert subsystem when decisions regarding adds, deletes, updates or queries must be processed. The rule-based expert subsystem must also be invoked when the data-base control subsystem encounters null values.

The system must allow the user the capability to add and delete rules from the rule-base. The user must also have the capability to override a decision made by the system. The system will provide explanations about all decisions made. The users will have the capability to establish a dialogue with the rule-based expert subsystem to aid in their research processes.

3.3 Specific Details

This section will present the specific details on the functional requirements of the Professional Genealogical Information System. The specific requirements can be subdivided into two categories corresponding with the two subsystems, data-base control and text manipulation, being addressed in this paper. Figure 3.1 depicts the integration between the subsystems of the Professional Genealogical Information System.

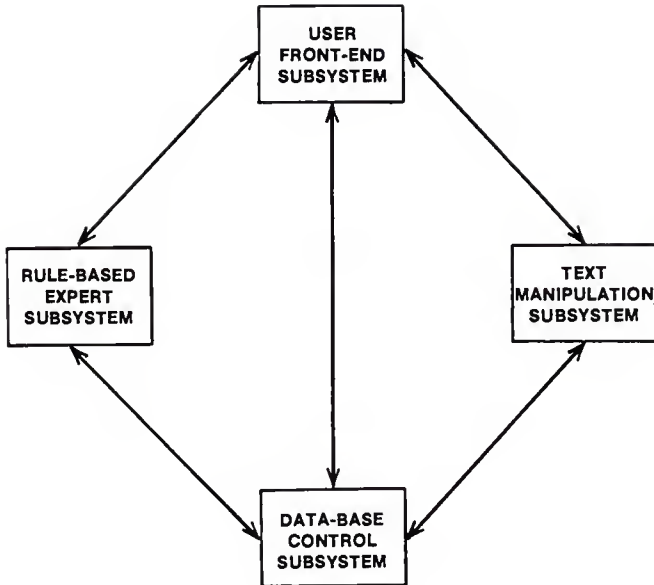


FIGURE 3_1. PROFESSIONAL GENEALOGICAL INFORMATION SYSTEM

3.3.1 Data-Base Control Subsystem

The Data-Base Control Subsystem as a whole will require a relational Data Base Management System (DBMS) suitable for use on a personal computer. A relational database approach has been chosen because it facilitates the formulation of nonprocedural, high-level queries, and this separates the user from the internal organization of the data. A hierarchical or network approach would involve more procedure oriented operations and would require the user to be familiar with the internal organization of the data. Even though the user will not be interfacing directly with the DBMS but instead with a user front end, it is still essential that the DBMS be as nonprocedural as possible in order to simplify the translation process between the user front end and the DBMS.

The data-base control subsystem will have to provide an Add capability so that the user can create and include new data in the database. The data that will be included in the database will be that which is collected regarding individuals being researched. The system will accept data from all types of source records. The DBMS will handle the insertion of the data as well as the establishment of any required links. If a decision is required before the Add can be completed the data-base control subsystem will

interface with the rule-based expert subsystem before completing the operation. Delete, Update and Query capabilities will also be required and their processing will need to be handled in the same manner as the Add.

Reporting capability will need to be provided by the system. The system must be capable of producing pedigree, descendant and family unit charts as output.

3.3.2 Text Manipulation Subsystem

The text manipulation subsystem must allow the user the capability of Creating, Deleting, Copying and Editing text. The system should provide Footnote capability to support the genealogist's historical documentation needs. Online command menus should be provided to support ease of use. The text manipulation subsystem should also allow the user the capability to create different formats such as lists, outlines, etc. Dictionary functions should also be provided to assist the user with spelling problems.

The text manipulation subsystem will need to allow the user the capability to navigate to the database so that data can be embedded directly into the text. This capability will require the text subsystem to interface with the data-base control subsystem.

CHAPTER 4 - DESIGN

4.1 Introduction

The Professional Genealogical Information System has been designed as four subsystems (as depicted in Figure 3_1): 1) the User Front End Subsystem (UFES), 2) the Rule-Based Expert Subsystem (RBES), 3) the Data-Base Control Subsystem (DBCS), and 4) the Text Manipulation Subsystem (TXMS). Figure 4_1 depicts an expansion of the four subsystems. This chapter will address the design of the DBCS and the TXMS; however, a general overview of the entire system will be presented first.

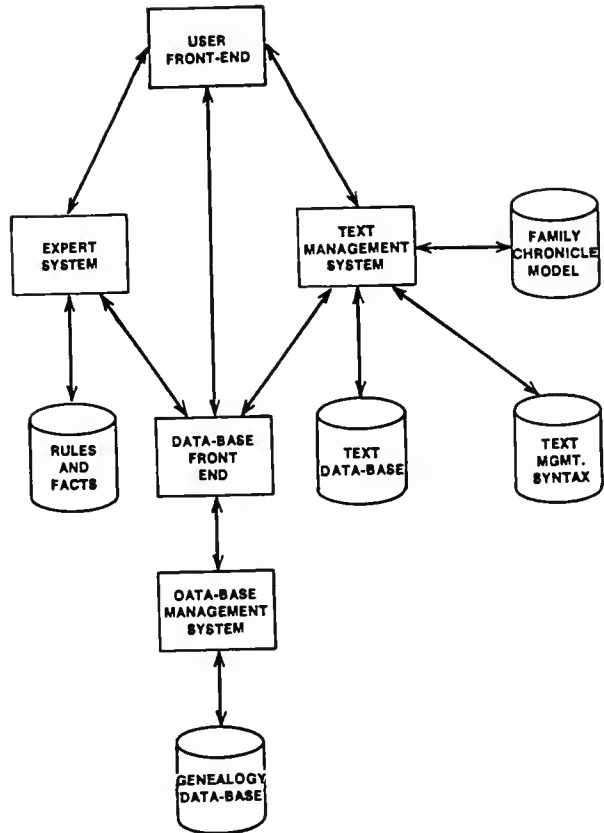


FIGURE 4.1. PROFESSIONAL GENEALOGICAL INFORMATION SYSTEM: EXPANDED VIEW

4.2 General Overview

Each of the four subsystems was designed to provide specific non-overlapping capabilities. The User Front End Subsystem was designed to provide a single interface to the system users. The User Front End Subsystem will provide command menus and preformatted screens to accomplish a dialogue with the users. The User Front End Subsystem will store the screen templates into a frame in memory so that the user entered data can be accessed by any of the other subsystems upon their invocation. The User Front End is responsible for determining which subsystem needs to be invoked based on the user's request. A user can request to add, delete or update rules from the rule base , as well as to establish a dialogue for research hypothesizing. These capabilities will be provided via the interface to the Rule-Based Expert Subsystem. The user can also request to add, delete, update or query data from the genealogy database. This capability will be provided via the Data-Base Control Subsystem interface. Finally, the user can request to create or edit text information associated with the development of family chronicles. This capability is provided via the Text Manipulation Subsystem interface.

The Rule-Based Expert Subsystem was designed to provide the Professional Genealogy Information System with a decision making capability. A rule base consisting of a set of packetized rules was developed with the aid of genealogy experts. These rules will be used each time a decision must be made. The design of the Rule-Based Expert Subsystem will not be presented in this paper since it is being addressed in another Master's thesis (BSP88) at Kansas State University. However, in discussing the design of the other subsystems the need to invoke the Rule Based Expert Subsystem for a particular decision will be presented. This should provide a clear understanding of the types of decisions that can be made by the Professional Genealogy Information System.

The Data-Base Control Subsystem was designed to provide the capability of storing, searching, comparing or extrapolating genealogy data. Genealogy data usually consist of a source record that provides descriptive information about a person or persons. A structured relational database has been designed as the storage mechanism for the genealogy data. The Data-Base Control Subsystem will be responsible for managing and controlling any processes that involve accessing the genealogy database.

The Text Manipulation Subsystem was designed to provide the capability of creating family chronicles. An unstructured text database has been designed as the storage mechanism for the family chronicles. Family chronicles are the major form of output produced by professional genealogists. In creating family chronicles, the genealogist must have readily available all information associated with each person within a family so that it can be included in the document. To provide simple access to the information stored about each person, the Text Manipulation Subsystem was designed to provide navigation capability from the unstructured text database to the structured genealogy database. This navigation capability will allow information obtained from the genealogy database to be directly embedded in the family chronicle. This will prevent the genealogist from having to access the database separately and then create the family chronicle. As far as can be determined from the research performed in this area, there are no other systems available on the market that provide a navigation capability from an unstructured database to a structured database. Although this capability is being designed for a genealogy application within this paper, it is a capability that is needed within various other applications.

4.3 Data-Base Control Subsystem (DBCS)

The Data-Base Control Subsystem (DBCS) has been designed with three components, the Data-Base Front End (DBFE), the Data-Base Management System (DBMS) and the Genealogy Database. Figure 4_2 depicts this design.



FIGURE 4_2. DATA-BASE CONTROL SYSTEM (DBCS)

4.3.1 Genealogy Database Design

Before discussing the Data-Base Front End and the Data-Base Management System, it is important to first discuss the design of the genealogy database. The Unger/Fisher design methodology, as described in Section 1.6 was used in this design process. This section will not depict the outputs from each phase of the design process but instead only those that help present an understanding of the database requirements.

Analysis of the source records used by genealogists to obtain information about individuals was performed and the required data elements were defined. Functional dependencies between the data elements were also defined. Entities (distinct objects within the enterprise), relationships (meaningful interactions between objects) and attributes (descriptive information about entities and relationships) were then derived for the genealogical enterprise. The Entity-Relationship diagram in Figure 4_3 depicts a semantic view of the genealogy enterprise.



FIGURE 4_3. ENTITY-RELATIONSHIP DIAGRAM

Figures 4_4 and 4_5 show an expansion of the Source and Person entities including their associated attributes.

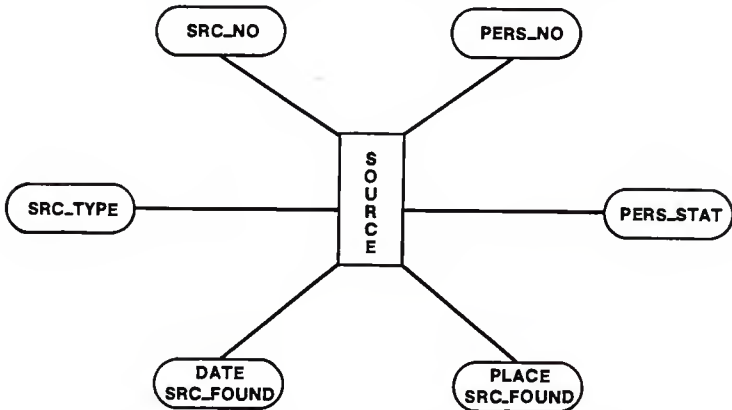


FIGURE 4.4. E-R SOURCE ENTITY

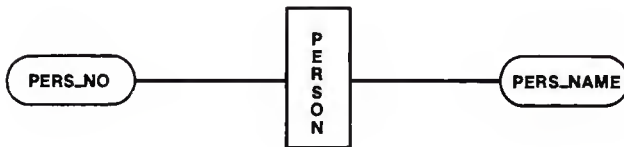


FIGURE 4.5. E-R PERSON ENTITY

All information collected about a person is obtained from a source record of one type or another. It is possible for each person to have multiple source records and for each source record to provide information about multiple people, hence the N-M relationship depicted in Figure 4_3.

When analyzing the "Provides Info" relationship depicted in Figure 4_3, it became apparent that there were actually different types of information that could be grouped together. Therefore, the "Provides Info" relationship should be expanded into multiple relationships. Figures 4_6 through 4_11 depict some examples of the actual relationship groupings. These groupings become extremely important to the genealogist when trying to determine precedence of one source record over another. For example, a person could have birth information provided from two or more sources and that information may differ somewhat. Therefore, the genealogist must know which source record takes precedence in order to determine accurate birth information.

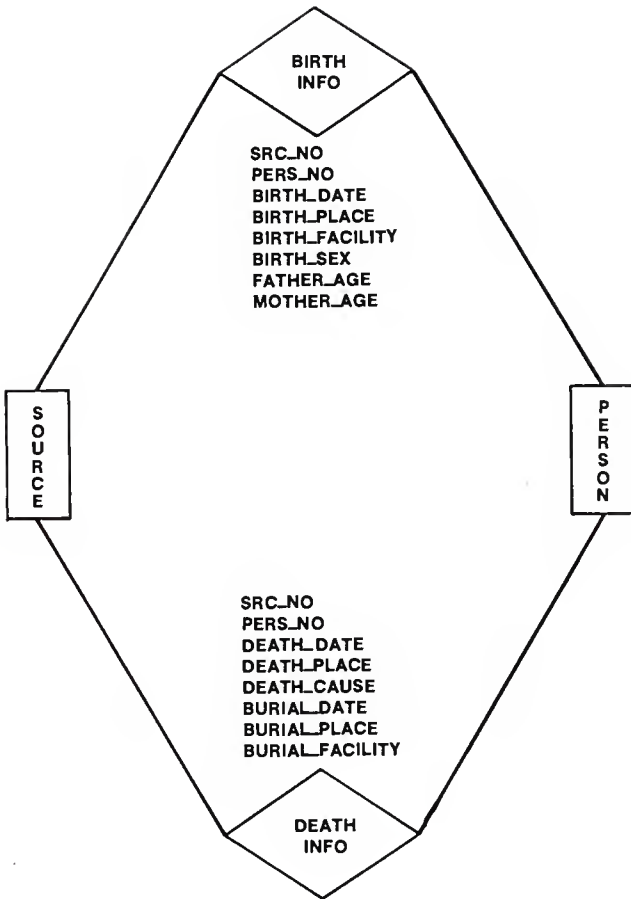


FIGURE 4.6. BIRTH AND DEATH RELATIONSHIPS

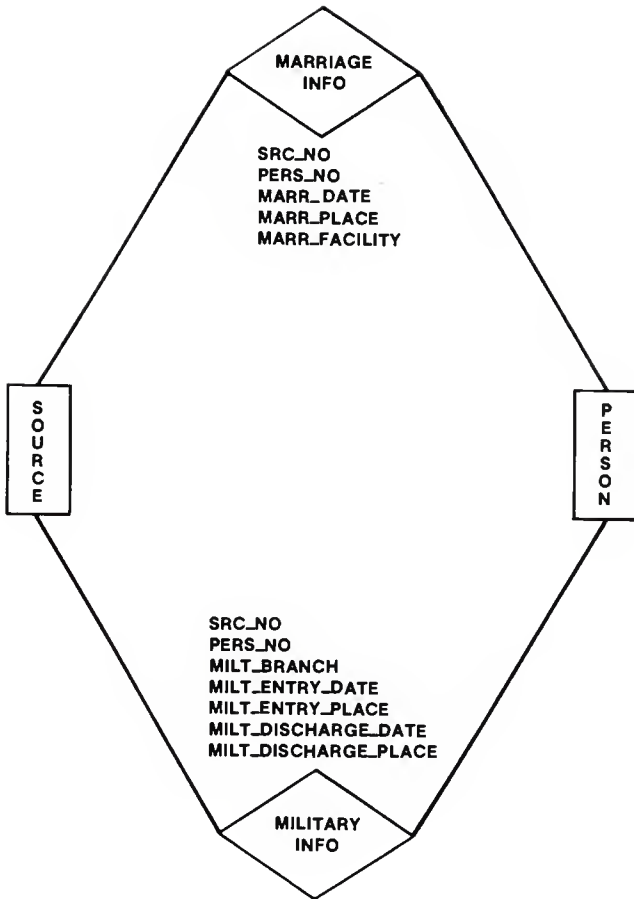


FIGURE 4.7. E-R MARRIAGE AND MILITARY RELATIONSHIPS

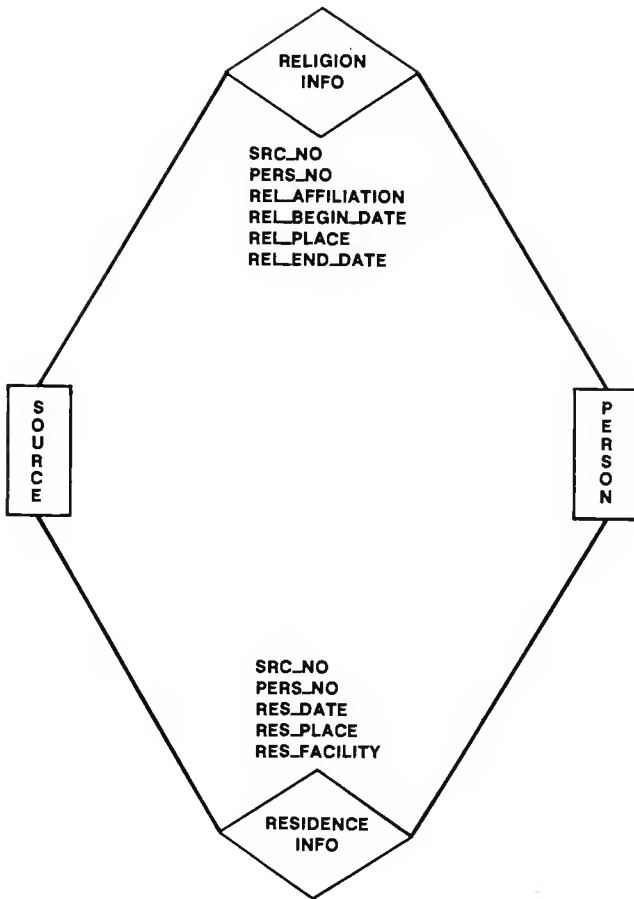


FIGURE 4.8. E-R RELIGION AND RESIDENCE RELATIONSHIPS

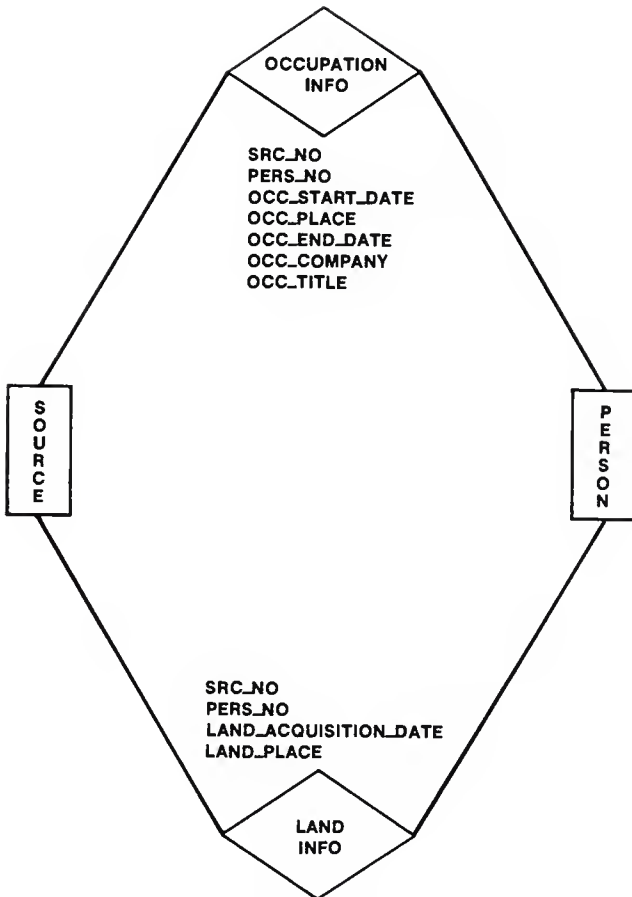


FIGURE 4.9. E-R OCCUPATION AND LAND RELATIONSHIPS

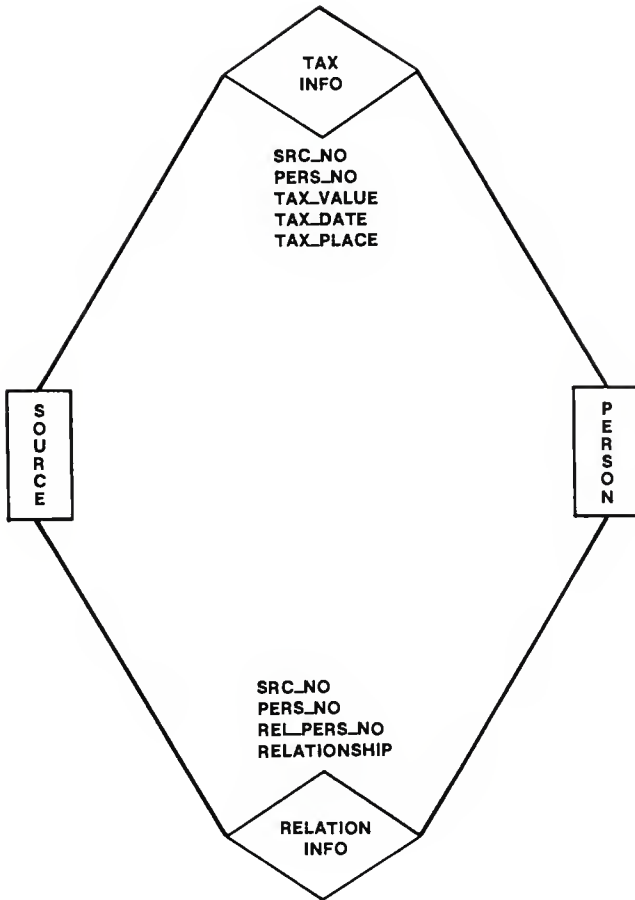


FIGURE 4.10. E-R TAX AND RELATION RELATIONSHIPS



FIGURE 4_11. E-R QUALIFYING RELATIONSHIP

A relational schema was then developed from the E-R diagram. Basically each entity and each relationship were transformed into a relation in the genealogical database. Table 4_1 presents the resulting relations.

Table 4_1 Relations

(SOURCE) Src_No, Pers_No, Src_Type, Pers_Stat,
Date_Src_Found, Place_Src_Found Key = Src_No,
Pers_No

(PERSON) Pers_No, Pers_Name Key = Pers_No

(PROVIDES BIRTH INFO) Src_No, Pers_No, Birth_Date,
Birth_Place, Birth_Facility, Birth_Sex, Father_Age,
Mother_Age Key = Src_No, Pers_No

(PROVIDES DEATH INFO) Src_No, Pers_No, Death_Date,
Death_Place, Death_Cause, Burial_Date, Burial_Place,
Burial_Facility Key = Src_No, Pers_No

(PROVIDES MARRIAGE INFO) Src_No, Pers_No, Marr_Date,
Marr_Place, Marr_Facility, Key = Src_No, Pers_No

(PROVIDES MILITARY INFO) Src_No, Pers_No,
Milt_Branch, Milt_Entry_Date, Milt_Entry_Place,
Milt_Discharge_Date, Milt_Discharge_Place
Key = Src_No, Pers_No

(PROVIDES RELIGIOUS INFO) Src_No, Pers_No,
Rel_Affiliation, Rel_Begin_Date, Rel_Place,
Rel_End_Date, Rel_End_Date, Key = Src_No, Pers_No

(PROVIDES RESIDENCE INFO) Src_No, Pers_No, Res_Date,
Res_Place, Res_Facility, Key = Src_No, Pers_No

(PROVIDES OCCUPATION INFO) Src_No, Pers_No,
Occ_Start_Date, Occ_Place, Occ_End_Date, Occ_Company,
Occ_Title, Key = Src_No, Pers_No

(PROVIDES LAND INFO) Src_No, Pers_No,
Land_Acquisition_Date, Land_Acquisition_Place
Key = Src_No, Pers_No

(PROVIDES TAX INFO) Src_No, Pers_No, Tax_Value,
Tax_Date, Tax_Place Key = Src_No, Pers_No

(PROVIDES RELATIONSHIP INFO) Src_No, Pers_No,
Rel_Pers_No, Relationship Key = Src_No, Pers_No,
Rel_Pers_No

(PROVIDES QUALIFYING INFO) Src_No, Pers_No,
Attribute_Name, Text_Addr, Prec_Flag Key = Src_No,
Pers_No, Attribute_Name

Most of the relations defined in Table 4_1 are fairly easy to understand; however, the "Provides Qualifying Info" relation should probably be discussed in more detail. This relation was designed to establish the capability to link unstructured text information with a structured data attribute value for which it is associated. This relation will also provide a means for denoting precedent data values for each data attribute defined in the genealogy database.

It is possible for each data value in the genealogy database to have additional qualifying text information for which it is associated. For example, if a birthdate is obtained from a source record on which the ink was smudged and the value was unclear, the genealogist may want to denote this situation in text format. The Text_Addr attribute will contain a file name (address) where the text information is stored and not the actual text itself. The need for this type of storage is discussed in more detail in section 4.3.2.

In addition, it is also possible for genealogists to obtain multiple data values for each data attribute. These data values can be identical or conflicting since they are normally obtained from different sources. Due to this type of situation, the "Qualifying Info" relation was

also designed to provide a means of denoting precedent values. The Prec_Flag attribute is simply a flag. If the attribute value is P, the associated tuple will provide the values needed to link to the actual precedent value. The attribute name and not the attribute value is stored in the "Qualifying Info" relation.

4.3.2 Data-Base Front End (DBFE)

One of the major objectives of the Professional Genealogical Information System is to provide simple to use storage, search and extrapolation capabilities. In order to meet this objective, it is necessary for the system to establish all identification numbers (person numbers, source numbers) and links between all related data. The user should be able to add, delete, update or query information for an individual based on the individual's name or a combination of name and other attributes that may be readily known. If users have this capability they do not necessarily have to worry with numbering schemes as is the case in the manual process. The system can always provide an indexed list of family linkages upon request. Since it is possible for there to be multiple people with the same name, the system must aid the genealogist in determining which person is needed.

The Data-Base Front End is needed to manage the invoking of the Rule-Based Expert Subsystem, the Text Manipulation Subsystem, and the Data-Base Management System as adds, deletes, updates and queries are received. The Data-Base Front End will receive add, delete and update requests directly from the User Front End. Query requests can be received from the User Front End, the Text Manipulation Subsystem or the Rule-Based Expert Subsystem, but each query is processed in the same fashion. When a user is requesting any of the database functions, the User Front End will provide preformatted screens to obtain any information needed. The screens will always be stored in a frame in memory so that the user entered data is readily accessible to the Data-Base Front End.

Since genealogists obtain all of their information from one source type or another, a request to add data to the genealogy database is equivalent to adding a source record. Therefore, when a user makes a request to add data, the User Front End always asks for the source record type to be entered. When the record type is selected, the User Front End provides a screen containing the standard fields for the record type chosen. The source owner's name from the source record is the minimum information the user must enter. The source owner is the person for which the record is primarily about. Secondary names can also

be obtained from a source record when the record type identifies relationships of secondary people to the source owner. For example, from John Doe's birth certificate, he is the source owner (primary person) and his father and mother are secondary people related to him. Once the user has entered all the fields for which values were available, the User Front End stores the screen in a frame in memory and then invokes the Data-Base Front End. A pseudo-code description of the processes performed by the Data-Base Front End when an add function is requested is contained in Table 4_2.

Table 4_2 DBFE Pseudo-Code Add Description

1. Obtain primary name from the UFES memory frame.
2. Generate a unique source number.
3. Invoke DBMS for query on NAME.
4. If match occurs, Then:
 - 4_a. Invoke RBES for decision on appropriate person.
 - 4_b. If person exists, Then:
 - 4_b_1. Obtain person number from RBES.
 - 4_c. Else:
 - 4_c_1. Generate unique person number.
5. Else:
 - 5_a. Generate unique person number.
6. Wait on Expert permission to Add.
7. If permission is not granted, Then:
 - 7_a. Reestablish any numbers generated as available for use.
 - 7_b. Terminate DBFE session.
8. Else:
 - 8_a. Invoke DBMS for Add
 - 8_b. Determine if secondary name exists in UFES memory frame.
 - 8_c. If secondary name exists, Then:
 - 8_c_1. Return to Step 3.
 - 8_d. Else:
 - 8_d_1. Terminate DBFE session.

The Data-Base Front End begins the add process by obtaining the primary name from the memory frame. The Data-Base Front End knows which attributes provide primary and secondary names for each source record type by accessing a table where these definitions are stored.

Step two of the add process requires the Data-Base Front End to generate a unique source number to be associated with the source record being entered. The Data-Base Front End is responsible for assigning and keeping track of all identification numbers.

Step three of the add process requires invoking the Data-Base Management System (DBMS) for a query on the primary name. The Data-Base Front End is responsible for establishing a query in the appropriate DBMS syntax. This query is issued to establish whether a person(s) with a name identical to the primary name already exists in the genealogy database. The DBMS query will request selection of the person number for any match on the primary name.

If one or more matches results from the query, step four of the process begins. Step four first involves invoking the Rule-Based Expert Subsystem for a decision as to which, if any, is the appropriate person. Upon receipt of the decision request, the Rule-Based Expert Subsystem may require more data about each person in question. The

Rule-Based Expert Subsystem can therefore request the Data-Base Front End to supply the data needed. The Data-Base Front End will issue appropriate queries (in DBMS syntax) to the DBMS to obtain any data needed by the Rule-Based Expert Subsystem. The Rule-Based Expert Subsystem will eventually return a decision. The decision will either indicate which person is appropriate or that none is appropriate and a new person should be created. If the appropriate person already exists in the database, the Rule-Based Expert Subsystem returns the associated person number. If the Rule-Based Expert Subsystem indicates a new person, the Data-Base Front End must generate a unique person number.

If no matches resulted following step three of the add process, the Data-Base Front End automatically assumes that a new person is to be created. This also results in the generation of a unique person number

The next step of the add process (step six), involves a wait state. Each time a user tries to add, delete or update a data attribute value, "demon-like" experts are activated. These experts check validity, integrity and precedence of the data values. It is possible for these experts to be active concurrent with the Data-Base Front End processing. Before the Data-Base Front End can

initiate a physical addition of any data to the genealogy database, the experts must complete their processing. Therefore, it is possible that the addition of data will be put on hold until permission to continue is either granted or not by the experts. The actual functions of the experts are not discussed in this paper but are explained in another Kansas State University Master's thesis (BSP88).

If permission from the experts is not granted, the Data-Base Front End performs step seven. This step involves reestablishing any identification numbers generated as available for use and then the Data-Base Front End session is terminated.

If permission from the experts is granted, the Data-Base Front End begins step eight. The DBMS will be invoked for an add (in the DBMS syntax) of all non-relationship type attribute values supplied by the user. Relationship type values are actually secondary people (such as, father and mother on a birth certificate) and links to the appropriate person numbers from the primary person number must be established when these type of values exist. Therefore, after addition of the non-relationship data is completed, the Data-Base Front End must determine if a secondary person has been specified. If there is a

secondary person, the Data-Base Front End returns to step three and begins the process again. The only difference in the processing of primary and secondary people is in the actual data added to the database. When processing a secondary person, the Data-Base Front End will invoke the DBMS for an add of a relationship between the primary and secondary people. The add for a secondary person does not result in linking all of the source data to the secondary person; however, the secondary person's relationship will be linked to the source number. This link is established so that it can always be determined from where the relationship information was obtained.

When the Data-Base Front End determines that all secondary people have been processed, the Data-Base Front End session is terminated.

Once the process for adding data values from a source record is complete, the User Front End Subsystem will ask the user if any qualifying text information associated with a data value needs to be added. If the user responds positively, the User Front End requests the user to highlight the data attribute involved. The user will then be asked to enter the qualifying text and to supply a file name for this text. This screen will once again be stored in a memory frame and the Data-Base Front End will

be invoked requesting a text add for the source record just processed. The Data-Base Front End will issue an add (in DBMS syntax) to the DBMS providing all linkage information as well as the file name under which the text is to be stored. The file name is inserted in the TEXT_ADDR attribute of the "Qualifying Info" relation described in Section 4.3.1. The Data-Base Front End will then invoke the Text Manipulation Subsystem to transfer the text from the memory frame to an available block of memory. The Text Manipulation Subsystem is responsible for managing the blocks of memory where qualifying text information is stored and labeling the beginning addresses with the file name provided. The capability to store text information outside of the database is provided to allow for variable length text information. If the text were stored directly in the TEXT_ADDR attribute, a maximum length would need to be defined. By storing a file name in the TEXT_ADDR attribute, the information can always be retrieved and the method of retrieval is transparent to the user. Handling qualifying text in this manner will also allow the user the capability to edit the text by providing the file name to the Text Manipulation Subsystem via the User Front End. If the user decides to delete this text, the User Front End will invoke the Data Base Front End requesting a delete of text information. The

Data-Base Front end will then in turn invoke the DBMS to delete the qualifying text link. The Text Manipulation Subsystem will then be invoked requesting a delete of the file name.

The key point here is that designing the system in this fashion provides the capability to navigate from a structured database to an unstructured database, which gives the user much more needed flexibility.

The procedures for processing deletes, updates and queries are basically the same as for the add. Therefore, a walk through of these processes will not be presented. The reader should be able to understand the processes by referencing the pseudo-code descriptions in Tables 4_3, 4_4, and 4_5 and by reviewing the add process.

Table 4_3 DBFE Pseudo-Code Delete Description

1. Obtain primary name from the UFES memory frame.
2. Invoke DBMS for query on NAME.
3. If match occurs, Then:
 - 3_a. Invoke RBES for decision on appropriate person.
 - 3_b. If person exists, Then:
 - 3_b_1. Obtain person number from RBES.
 - 3_c. Else:
 - 3_c_1. Generate error message to UFES.
 - 3_c_2. Terminate DBFE session.
4. Else:
 - 4_a. Generate error message to UFES.
 - 4_b. Terminate DBFE session.
5. Wait on Expert permission to Delete.
6. If permission in not granted, Then:
 - 6_a. Terminate DBFE session.
7. Else:
 - 7_a. Invoke DBMS for Delete.
 - 7_b. Terminate DBFE session.

Table 4_4 DBFE Pseudo-Code Update Description

1. Obtain primary name from the UFES memory frame.
2. Invoke DBMS for query on NAME.
3. If match occurs, Then:
 - 3_a. Invoke RBES for decision on appropriate person.
 - 3_b. If person exists, Then:
 - 3_b_1. Obtain person number from RBES.
 - 3_c. Else:
 - 3_c_1. Generate error message to UFES.
 - 3_c_2. Terminate DBFE session.
4. Else:
 - 4_a. Generate error message to UFES.
 - 4_b. Terminate DBFE session.
5. Wait on Expert permission to Update.
6. If permission in not granted, Then:
 - 6_a. Terminate DBFE session.
7. Else:
 - 7_a. Invoke DBMS for Update.
 - 7_b. Terminate DBFE session.

Table 4_5 DBFE Pseudo-Code Query Description

1. Obtain primary name and attributes in question from the UFES memory frame.
2. Invoke DBMS for query on NAME.
3. If match occurs, Then:
 - 3_a. Invoke RBES for decision on appropriate person.
 - 3_b. If person exists, Then:
 - 3_b_1. Obtain person number from RBES.
 - 3_c. Else:
 - 3_c_1. Generate error message to UFES.
 - 3_c_2. Terminate DBFE session.
4. Else:
 - 4_a. Generate error message to UFES.
 - 4_b. Terminate DBFE session.
5. Invoke DBMS for query on attributes of person number supplied.
6. Return results to UFES.
7. Terminate DBFE session.

A delete request will involve the deletion of either a person, an entire source record for a person or individual data attributes for a person. At any time a delete is encountered the DBMS must "clean-up" all affected links. An update request will involve changing the value of a particular data attribute(s) for a person. An update should not involve changing an entire source record, this should be accomplished with a delete and a subsequent add. As with the add and delete any changes that affect links will have to be "cleaned-up" by the DBMS. A query request will involve obtaining any data that matches that of the attribute(s) specified by the user.

4.3.3 Data-Base Management System (DBMS)

A relational model DBMS has been chosen in which to implement the Professional Genealogical Information System. It is important that the DBMS have a user exit in order to provide the capability of interfacing with other application programs such as the Data-Base Front End. The DBMS would have to be programmed to provide insertion, deletion, update, query and linkage capabilities for the data attributes in the Genealogy Database.

4.4 Text Manipulation Subsystem (TXMS)

One of the objectives of the Professional Genealogical Information System is to provide text manipulation capability. This capability is needed to aid genealogists in the creation of family chronicles. The creation of family chronicles is currently a difficult process because it requires genealogists to have readily available the descriptive information about each individual being documented. The Text Manipulation Subsystem will address this problem by providing the capability to navigate from an unstructured text database, where the family chronicle is stored, to a structured genealogy database where the descriptive information is stored. By having this navigation capability, descriptive information can then be embedded directly into a family chronicle upon its creation. To provide this type of capability, the Text Manipulation Subsystem has been designed with four major components: 1) A Family Chronicle Model, 2) A Text Management Syntax, 3) A Text Management System and 4) A Text Database. Figure 4_12 depicts the interfaces between these components.

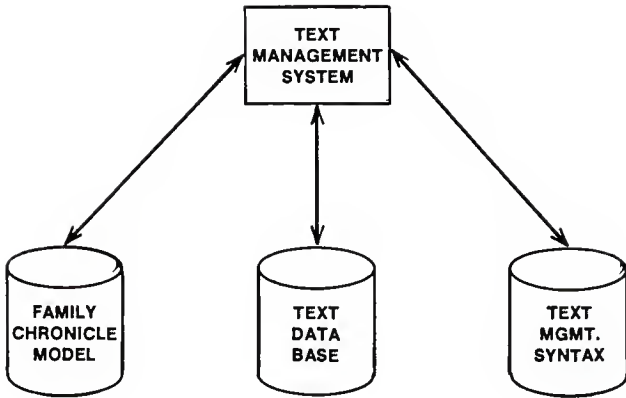


FIGURE 4_12. TEXT MANIPULATION SUBSYSTEM COMPONENTS

4.4.1 Family Chronicle Model

A model of the typical structure of a family chronicle must be developed to provide a basis for determining where descriptive information should be embedded. The development of such a model would require extensive communication with genealogy experts, which exceeds the scope of this paper. However, a general design recommendation will be presented and used when discussing the other components of the Text Manipulation Subsystem.

The Family Chronicle Model should define some feasible structure, such as a paragraph, for each type of descriptive information possible for any person. The types of descriptive information should basically correspond to the relations (Birth Info, Religious Info, Death Info, etc.) defined in Table 4_1. The model should also define an order for the information type structures, i.e., birth information about a person is first, religious information is second, occupation information is third and so on until all information types are presented in a reasonable order.

Within the information type structure another structure, such as a sentence, should also be defined. Within the sentence structure an order and position of descriptive

keywords should be defined. The position of the keywords will eventually determine the position for embedding data values obtained from the genealogy database. Keywords used in the model could basically correspond to the data attributes defined in the information relations in the genealogy database. An overall structure, such as a chapter, that encompasses the paragraph and sentence structures should be defined for each person being documented in the family chronicle. To clarify this design even further, Figure 4_13 provides a representation of the model.

Chapter 1 - NAME

Paragraph 1 - Birth Information

NAME was born on BIRTHDATE in BIRTHPLACE.
The birth occurred at BIRTHTIME in
BIRTHFACILITY. FATHERNAME was the father
at age FATHERAGE and MOTHERNAME was the
mother at MOTHERAGE.

.
.
.

.
.
.

Paragraph N - Death Information

NAME died on DEATHDATE at DEATHPLACE.

. . .
.
.
.

Chapter N - NAME

Figure 4_13 Family Chronicle Model

The model depicted in Figure 4_13 shows Birth Information with its descriptive keywords as the first type of information presented and Death Information with its descriptive keywords as the last type of information presented for each person being documented. An order such as this should be established and each person documented would use this overall structure.

4.4.2 Text Management Syntax

In order to use the Family Chronicle Model, a Text Management Syntax that can translate the structure types defined in the model must be developed . The syntax would have to define the keywords needed and the positioning of those keywords within each of the structure types. That is, the keywords defined would have to occur in a certain position within a certain sentence within a certain paragraph for each chapter created for a person. The number and order of the chapters would be totally dependent on the number of people the user wants to document within one family chronicle. Figure 4_14 depicts an informal syntax description that corresponds to the model in figure 4_13.

Chapter N - Title = NAME(5,50c) {PERS_NO(6,6a)}

Paragraph 1 - Title = Birth Information

Sentence 1 = Keyword 1 = NAME(1,50c)
Keyword 2 = BIRTHDATE(5,dd/mm/yy)
Keyword 3 = BIRTHPLACE(7,50c)

End Sentence 1

Sentence 2 = Keyword 1 = BIRTHTIME(5,hh:mm:ss)
Keyword 2 = BIRTHFACILITY(7,20c)

End Sentence 2

Sentence 3 = Keyword 1 = FATHERNAME(1,50c)
Keyword 2 = FATHERAGE(7,2a)
Keyword 3 = MOTHERNAME(9,50c)
Keyword 4 = MOTHERAGE(15,2a)

End Sentence 3

End Paragraph 1

.

.

.

Paragraph N - Title = Death Information

.

.

.

End Paragraph N

End Chapter N

- * Underlined words represent structure types.
- * Words in all capital letters are keyword names which correspond to attribute names in the genealogy database
- * Information inside parenthesis represents (position, format).
- * Information inside braces indicates that it is not visible to the user and is for system use only.
- * The "N" will have to be maintained by the system.

Figure 4_14 Text Management Syntax

The development of a formal syntax, like the creation of the model, is beyond the scope of this paper; however, it is important to understand the design philosophy so that the functions of the Text Management System can be better understood.

4.4.3 Text Management System - Text Database

The overall purpose of the text management system is to allow the user to create family chronicles in which the system will insert descriptive data values from the genealogy database.

When the Text Management System is invoked by the User Front End, a user has made a request to establish a text session and has identified the type of session preferred. There are basically four types of text sessions available to the user: 1) create a new family chronicle, 2) create a new person within an existing family chronicle, 3) create a new information structure for an existing person within an existing family chronicle or 4) edit an existing family chronicle.

When session type one is specified, the User Front End will request that the user supply a unique file name for the chronicle and the name of the first individual the user wants to document. When the User Front End invokes the Text Management System, the user entered data is

available from a frame in memory where the screen was stored by the User Front End. The Text Management System must then invoke the Data Base Front End requesting a person number for the NAME obtained from the memory frame. After receipt of that number, the Text Management System will access the Family Chronicle Model to determine the first structure type needed.

The first structure at the beginning of any chronicle will be a chapter. The defined syntax will then have to be used to determine the keywords, their positions and their formats within this structure. In the case of the chapter, there are only two keywords, NAME and PERS_NO, and those values are already available to the Text Management System. Therefore, the Text Management System will copy the chapter structure from the model into the text database and label the beginning address with the filename specified by the user. The Text Management System will then replace the keyword NAME, at the appropriate position in the chapter structure, with the person's name. The name will be stored in the format specified by the syntax. The person's number obtained from the Data-Base Front End will also be stored with the name but this value will not be visible to the user.

At this point, each of the following structures must be obtained one at a time from the Family Chronicle Model. For each structure obtained, the syntax must be used to determine the associated keywords. The keyword names defined in the syntax will correspond directly to the attribute names used in the genealogy database so that no translation is necessary to obtain values. When the values for the keywords are not readily available, unlike the case with NAME and PERS_NO, the Text Management System must invoke the Data-Base Front End requesting the precedent values for the attributes involved. Once the values are obtained the structure in question must be copied into the text database. The values obtained will then replace their associated keywords in the format specified by the syntax. If no value is obtained for a particular attribute, the keyword name will remain in the text. If no values exist for any of the attributes, the structure in question will not be copied to the text database. After this process is complete, the Data-Base Front End must be invoked again to obtain any qualifying text data and all non-precedent values and their associated record. All this type of information will be appended in free form style immediately following the structure type just created. This information is included to provide the user with additional crucial facts.

The process of obtaining each structure one at a time, obtaining the data values for the associated keywords and inserting the structure and values in the text database is repeated until the end of the Family Chronicle Model is reached. At this point, the Text Management System returns control to the User Front End. The User Front End is then responsible for accessing the family chronicle file just created and transferring it to the user's screen for viewing. The user can then choose to exit the text session or choose another session type.

When a user elects to add a person to an exiting family chronicle (session type two), the processes specified above are followed and the text created is appended to the existing chronicle - beginning a new chapter. The Text Management System is responsible for recording the number of people in a chronicle so that appropriate chapter numbers can be determined. Data of this nature will be stored at the beginning of each family chronicle file within the text database, but it will only be accessible by the system.

If a user realizes that a new type of information was entered into the genealogy database after the creation of a person's structure within a family chronicle, the user may want to establish a text session to add an information

structure to an existing person in an existing family chronicle (session type three). Upon receiving a request like this, the User Front End will once again request that a filename and an individual's name be supplied. In addition the user will also be requested to select the information type to be added. Once this information is obtained and stored in the memory frame, the Text Management System is invoked. The Text Management System will retrieve the filename specified and search each chapter structure for the name specified. This search capability will have to be performed with the aid of the syntax. If multiple names are found that match the name provided by the user, the Text Management System will have to return control to the User Front End requesting identification of the appropriate person. The User Front End will then transfer the file to the user's screen and request the user to select the appropriate chapter number. Once this selection occurs, it is stored in the memory frame and control is returned to the Text Management System. From this point, the Text Management System must find the user specified information structure in the Family Chronicle Model. After the structure is obtained, the process followed is identical to that described for the two previous session types. The text that is created

is inserted in its appropriate position within the file according to the model's established order.

Once a family chronicle has been created, a user will more than likely want to edit the structure (session type four). This editing capability will allow the user to add, delete, move, expand, etc. any portion of the file specified. Basically the user can use this capability to mold the family chronicle into a format he/she prefers. For example, the qualifying information provided for each paragraph structure could be formatted as footnotes for its associated data value in the text, or the user could elect to delete the information entirely.

4.5 Summary

The major goal of the design effort described in this chapter was to provide the capability to navigate from an unstructured database to a structured database and vice-versa. This capability was presented for a genealogy application in this paper; however, it is a capability that could be incorporated into other applications. The development of such a navigation design could well further the integration of Office Information Systems and Database Management Systems.

CHAPTER 5 - IMPLEMENTATION

5.1 Introduction

The implementation presented in this chapter concentrates on one subsystem of the overall design discussed in Chapter Four. The implementation addresses a portion of the Data-Base Control Subsystem and more accurately the Genealogy Database and the Data-Base Management System.

The system, PIGS (Professional Information Genealogical System), was implemented using dBASE III PLUS on an AT&T 6300 personal computer. dBASE III PLUS was chosen due to the requirement to provide a relational database management system capable of running on a personal computer. Also, dBASE III PLUS provided the capability to have multiple attributes as a key expression for an index file, to have multiple index files active at one time, and to establish relations between multiple database files. These characteristics are important due to the complexities associated with Genealogy. Facts such as, multiple people can have the exact same name, one person can have multiple source records of the same type and multiple people can be identified from one source record are just a few of the reasons for the complexities. These complexities were taken into account in the design process of the Genealogy Database.

5.2 Genealogy Database Structure

The Genealogy Database consists of multiple database and index files. Each entity and relationship defined in Chapter Four was transformed into a database file (relation). The structures of the database and index files implemented for PIGS are depicted in Tables 5_1 and 5_2.

Table 5_1 Database File Structures

Database File	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
	COUNTY_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_PERS_NO	Character	6
	RELATION	Character	10

Table 5-1 Database File Structures (Cont.)

Database File	Field Name	Type	Width
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	Date	8
	BURY_CITY	Character	20
	BURY_CNTY	Character	20
	BURY_ST	Character	2
BURY_FAC	Character	10	
MARRIAGE_IN	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 5_2 Index File Structures

Database File	Index File	Key Expression
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
SOURCE	PERSTYPE	PERS_NO+SRC_TYPE
	SRC	SRC_NO
	SPERS	PERS_NO
RELATION	RSRC	SRC_NO
BIRTH_IN	BSRC	SRC_NO
	SRCPERS	SRC_NO+PERS_NO
DEATH_IN	DSRC	SRC_NO
	DSRCPERS	SRC_NO+PERS_NO
MARRIAGE_IN	MSRC	SRC_NO
	MSRCPERS	SRC_NO+PERS_NO

The database files created follow the design with the exception of the PRSNO and the SRCNO.database files. These database files were never directly discussed but were established to provide a means for creating, storing and re-utilizing identification numbers for people and source documents.

The index files were created based on the type of access required for manipulating the data stored in these files. That is, the index files are simply a means of efficiently accessing data in a database file by particular data attributes (key expressions) based on the data values most commonly used by the PIG System.

5.3 Data-Base Management System - PIGS

The PIG System provides the functionality for genealogists to add, delete, update, link, query and report data. The system is source record driven. All functions performed by the system are done via a particular source record, with a source record being a Birth Certificate, Marriage Certificate or Death Certificate. The system insures this type of performance by providing a simplistic user interface to guide the genealogist in activating any of the functions mentioned.

5.3.1 User Interface

The User Interface consists of menu driven screens that allow the user to select particular options. The Main Menu screen requires the user to select which function (add, delete, update, query or report) they want to perform. This then activates the Source Record Screen which requires the user to select the source record upon which they want the function to be performed. Next the Data Screen is activated which requires the user to specify particular data values associated with the function and source record previously selected. The actual screen information can be obtained from the code provided in Appendix C.

5.3.2 Add Function

The Add Function allows genealogists to add source records. When a source record is added in association with a person(s) that does not already exist in the database, the person(s) is also added. The addition or existence of a person is determined by the user with guidance from the system. The system provides information to the user about every person(s) that has the same name(s) as entered in any name field in the source document. The user is then asked to select the appropriate existing person(s) or to establish a new

person(s). If a new person is specified, a unique person identification number is generated and associated with that person. Since, typically there is always information about a primary and one or more secondary people on each source record, multiple people can be added from the addition of one source record. When a secondary person is entered on a source record, the system will insure that all appropriate links between the primary and secondary people are established.

The addition of a source record requires that the user supply a minimum of a name and date associated with the primary person. The values acceptable for the name are: last and first names; last and middle names; or last, first and middle names. The values acceptable for date are any birth, marriage or death dates prior to the current date. The system checks the validity of these entries.

A record is entered in the Source Database File for each primary and secondary person specified. This allows the user to track where information was obtained about each person. For every source record added, the system generates and associates a unique source identification number. If new people were added, an entry into the Person Database File for each individual also occurs.

Entries in the Relation Database File are recorded to establish links and types of relationships between primary and secondary people. Finally, entries into the appropriate information database (Birth_In, Marriage_In or Death_In) are also recorded to store pertinent birth, marriage or death data.

5.3.3 Delete Function

The delete function allows the user to delete a source record. The user is required to specify the name of the primary person on the source record. The system will supply information about each person in the database with the same name as specified. The user then must select the appropriate person and source record to be deleted.

If the primary person specified does not have any other source records (besides the one being deleted), then the person is also deleted from the database. The deletion of a source record will always result in the deletion of the Source Database entries, the Relation Database entries and the Birth_In, Marriage_In or Death_in Database entries. This insures that all information and links obtained from the source record are eliminated. When the data is deleted, the source record identification number is re-established as available for use. If a person was also deleted as a result of the source record deletion, then

the person identification number is also re-established as available for use.

In order to provide efficient access to all data associated with a source record being deleted, the system has four work areas established with a different database file and its appropriate index file(s) active in each. In dBASE III PLUS with this type of implementation, it is then possible to establish relations between the different database files. The relations are established using key expressions. What this provides is an efficient way to access all related data once a value for the key expression has been found in one database file. This avoids the need to search for the key value in each database file separately since a pointer is moved to the appropriate record matching the key expression value in each active database file for which a relation was established. The only limitation here is that only one relation can be established between any one active database file to another.

5.3.4 Update Function

The update function allows the user to update a source record. This function is provided so that additional values for fields that were not previously entered can be included and typing errors can be corrected. Once again,

this function also requires the user to supply the name of the primary person for which the source record is about. The update function then performs as did the delete except that it will supply the user with the source record and values that were included when it was added. The final result will be replacing fields that are changed by the user with the new values entered. The same techniques for relating active database files and their index files were used in implementing the update function.

5.3.5 Query/Report Function

The query function allows the user to query information in the database. The queries that have been implemented are the Source Record Query and the Lineage Query. The source record query provides the user with a particular source record for a particular individual. This does not differ from the update function except that the user is not allowed to change any data during a query request.

The lineage query provides the user with either an ancestral or descendant lineage. In implementing this function there was a need to establish precedence rules. These rules were needed since it is possible, for example, for a person to have a different Mother's name on their Birth Certificate than is on their Marriage Certificate. There are legitimate reasons for situations like this to

occur, for example, a person's biological mother could have died and by the time the person was married, they had been adopted and the adopting mother's name was included on the Marriage Certificate. Since the biological lineage is the one of importance in genealogy, it was decided that a Birth Certificate would take higher precedence over the other two. The remaining order is Marriage Certificate then Death Certificate. This final order was decided based on the fact that if a person does get married they normally do it before they die, therefore, the likelihood that the Marriage Certificate would be more accurate. Since people can be included in the database as a result of being a primary or secondary person on a source record, it was also essential to create some intelligence in the Source_Type value that was entered when the record was added. The system, therefore, depicts the source record type for a primary person as XC and for a secondary person as XR, where X is either B (Birth), M (Marriage) or D (Death).

When creating an ancestral lineage, the system searches for the XC source record types, looking for the BC first. If there are multiple BCs for the person, the system chooses the first one in the index list. There is no predetermined precedence established for multiple entries of the same source record type since this would require

some detailed analysis and intelligence to be built into the system. In chapter four, this intelligence was included as part of the Rule-Based Expert Subsystem design. Once the appropriate source record is found for the primary person, the Relation Database File is used to find the identification numbers for the father and mother (secondary people). Then the father and mother are processed in the same manner as was the primary person until no more links can be found.

When creating a descendant lineage, the system searches for the XR source record types, looking for the BR first. Once the source record is found, for the secondary person in this case, the Relation Database is used to find the identification number for the child (primary person). This information is stored and all other XR source records are evaluated looking for multiple children. Each child's identification number is stored and when all children are found, each child is processed as was the parent.

The two queries discussed above are also available as reports. The user is allowed to request that the query information be transported to a printer. No additional reports were developed for this implementation.

5.4 Summary

The implementation presented in this chapter is that of a prototype-type of the Professional Information Genealogical System (PIGS). This system is a relational genealogy database system developed for use on a personal computer. The system is source record driven since a source record is always required by genealogists to derive information. Three major types of source records, Birth Certificate; Marriage Certificate and Death Certificate, were provided in this implementation.

The system has a simplified user interface to guide even the novice computer user through the appropriate steps. The user interface system is menu screen driven and allows the user to select particular options or supply specified data values.

The system allows the user to add, delete, query and report data based on individuals' names instead of on identification numbers. The system does generate and use identification numbers, but the user is not required to remember or track these numbers. The identification numbers are a necessity to provide uniqueness and normalized databases. The system also automatically establishes the appropriate linkage of people and source information so that genealogists can obtain detailed

lineage information.

PIGS is a prototype-type system since limited source records and reports were actually implemented. The system did involve detailed design and development work due to the complexities associated with genealogy itself. The code was written in dBASE III PLUS and is contained in Appendix C.

CHAPTER 6 - CONCLUSIONS AND EXTENSIONS

6.1 Introduction

The conclusions and extensions presented in this chapter will be discussed in two separate sections. The first section, 6.2, will discuss conclusions and extensions associated with the requirements (Chapter 3) and design (Chapter 4) of the overall Professional Information Genealogical System. The second section, 6.3, will discuss conclusions and extensions associated with the implementation (Chapter 5) of the Genealogy Database and the Data-Base Management System.

6.2 Conclusions and Extensions for the Requirements and Design

The requirements and design discussed in this thesis are of a very complex nature and would require some sophisticated personal computer tools to implement. An Artificial Intelligence Tool would be needed to develop the Rule-Based Expert Subsystem, a Word Processing Tool would be needed to develop the Text Manipulation Subsystem, a High Level Programming Language would be needed to develop the User Front-End Subsystem and a relational Data-Base Management Tool would be needed to develop the Data-Base Control Subsystem. To find the tools needed to develop the four subsystems and also

provide interfacing between all four would prove to be a difficult task on a personal computer

The Text Manipulation Subsystem could probably be developed using a Word Processing System for a personal computer. There are several word processing products available today that provide exits to user written applications. The Word Processing System itself could be used to manipulate the Text Database while the user written application (Text Management System) would have to control manipulation between the Family Chronicle Model, the Text Management Syntax and the Word Processing System (Text Database). In addition the user written application would have to interface with the other three subsystems. This indicates that it would probably be more effective if the Text Management System and the other three subsystem interfaces were developed jointly and in the same programming language. The possibility of finding existing personal computer tools that would address the needs of each subsystem is not unlikely; however, the possibility of having all of these tools interface with each other is a different story. It seems that to develop the overall system it would be more practical and efficient to develop each tool and all of the interfaces in a high level language, possibly even C Language. In order to perform this development task several man years would have to be

spent in developing the artificial intelligence techniques and the unstructured to structured database navigation techniques, let alone the overall interfacing required between each subsystem. It would be more effective to develop such a system using existing relational database management, artificial intelligence and word processing tools so as to avoid re-inventing of the wheel. But once again ,the practicality seems extreme.

In summary, it appears that the design is solid but the problem comes in attempting to develop such a product on a personal computer. Each subsystem in itself would require a massive amount of memory and processing time not to mention what would be required if all four subsystems were running together as an overall system. Any development of such a system for a personal computer would have to be efficient in all areas in order to provide its users with reasonable response times. The development of a system to address only the navigation between unstructured and structured databases could probably be accomplished with the aid of word processing and data-base management products that provide user exits. As personal computer products (hardware and software) become more and more advanced, the task of developing the design discussed in this thesis may become trivial; but, with the products available on the market today, it does not appear to be a

trivial task.

Extensions associated with the requirements and design specified in this thesis could include more research on existing personal computer hardware and software tools. The requirements and design could then be re-established based on the capabilities and limitations of the tools. As they were written in this thesis, however, thought was only given to the general types of tools needed but not to the specific tools themselves.

6.3 Conclusions and Extensions for the Implementation

The implementation presented in this thesis is a prototype of the Genealogy Database and the Data-Base Management System. The system was developed on an personal computer using dBase III Plus, a relational database management system. Some limitations associated with dBase III Plus made the implementation more cumbersome than would have been necessary if another tool had been available. Although dBase III Plus is a fairly advanced relational database management tool for personal computers, it does not provide the capability to: 1) have more than seven index files active at on time, 2) have multiple relations established simultaneously between two database files or 3) have simplistic use of non-character attributes as key

expressions for index files

6.3.1 Activation of Multiple Index Files

In order to provide the capability to efficiently search on more than just a name attribute provided by the user , would require that more index files be established and active for each database file. It proved difficult to limit the number of index files to seven in the prototype implementation, and it only allows for the use of a name attribute for searching. This situation occurred since the prototype did allow for one of three combinations of names to be provided.

6.3.2 Relating Multiple Database Files

Only one relation can be defined or active from one database file to any other database file. Relations are established using key expressions of index files. This requires the database files to be indexed on key attributes needed to establish relations. This presents a need for active index files beyond the search requirements. Once again, the limitation of only seven active index files makes this implementation more cumbersome.

Since only one relation can be defined or active between any one database file and another, it becomes necessary to

chain relations between database files. For example, to relate the Source Database File with the Birth_In Database File and the Relation Database File using a key expression of SRC_NO+PERS_NO would require that a relation be established from Source to Birth_In and that a relation then be established from Birth_In to Relation, with both relations using the same key expression. If another key expression (index file), besides the one needed to establish the relations, is required to find the correct record, as is the case in the prototype system, then multiple seeks must be performed before the appropriate linkage between the database files can be accomplished.

6.3.3 Non-Character Key Expressions

A key expression for an index file can be one database field or a combination of fields. When a combination of fields is used, the concatenation is denoted by means of a plus (+) sign. If concatenation were truly the meaning of this denotation, there would be no problem using different field types to establish key expressions. However, the actual interpretation of this denotation by dBase III Plus is somewhat different. If the key expression consists of multiple character type fields, dBase III Plus does present what is expected in the index file. However, if the key expression consists of multiple numeric type

fields, dBase III Plus actually adds the values of the fields and then establishes the index file. Obviously, this can and will present anomalies in the database. In order to avoid these anomalies, the system must be developed using only character type key expressions or conversion of all key fields to character must occur. There are functions available in dBase III Plus that provide simplistic conversion, but the problem arises when trying to use those fields in a search process throughout the system. Each time it is required to search for a value that is numeric, the conversion function must be performed so that the value matches that of the converted key expression in the index file. This type of implementation leaves open the possibility for error and cumbersome debugging. Therefore, it is recommended that all fields used as key expression be of the character type. This type of implementation is not always possible, since often numeric values are required to perform arithmetic operations. If this is the case, then a conversion from character to numeric must occur and once again leaving open the possibility of error and anomalies.

Another limitation associated with key expressions for index files is that of multiple field types. dBase III Plus does not allow key expressions to consist of different field types. That is, a key expression cannot

consist of a character and a numeric field. Once again, in order to provide this type of implementation, a conversion on one of the field types would have to be performed and remembered when using that field throughout the system.

6.3.4 Summary of Conclusions and Extensions for the Implementation

Based on the limitations discussed in the previous sections, it is obvious that one extension would be to develop the prototype system with another database management tool. Future releases of dBase III Plus are suppose to relieve some of these limitations and would provide for a more extensive implementation. The prototype, as a result of the limitations encountered, only addressed three modified source records. An extension to the system would allow for more information to be collected from each source record as well as for more source records to be included. Another extension could involve providing the capability to search for information based on fields other than name or in combination with name. If the limitations were corrected, more effort could be concentrated on developing complex precedence rules which would in turn supply users with more built-in intelligence and assistance in their decision making processes.

6.4 Summary

Based on the facts presented in this chapter, the overall conclusions and extensions reside in finding advanced personal computer hardware and software tools capable of implementing such a complex design. The availability of more advanced tools may require re-design but would provide the capability of implementing a more effective system.

REFERENCES

- [LA85] Albright, L. Personal interview. 10, November, 1985.
- [LA&FML80] Albright, L. and Leary, F. H. M. "Research Strategies." in North Carolin Research Genealogy And Local History. eds., H. F. M. Leary and M. R. Stirewalt, Saline, M.I.: McNaughton and Gunn, 1980, pp.1-39
- [PAB76] Berstein, P. A. "Synthesizing Third Normal Form Relations From Functional Dependencies. ACM Transaction on Database Systems 1(1976): 277-298
- [LB&JPG86] Bic, L. and Gilbert, J. P. "Learning From AI: New Trends In Database Technology." ComputerE, (March 1986):44-54
- [CJD86] Date, C. J. An Introduction In Database Systems Reading, M.S.: Addison Wesley Publishing Company, 1986
- [ITH84] Hawryskiewicz, I. T. Database Analysis And Design Chicago, I.L.: Science Research Associates, Inc., 1984
- [DK83] Kroenke, D. Database Processing Chicago, I.L.: Science Research Associates, Inc., 1983
- [FMHL86] Leary, F. H. M. Personal interview. 19, May, 1986
- [JM76] Martin, J. Principles of Data-Base Management Englewood Cliffs, N.J.: Prentice - Hall, Inc., 1976
- [GM] Moody, G. "Lotus Report Writer, Add On For 1-2-3 And Symphony." Personal Computing
- [MA86] Anthony, M. "When You Should Switch To A Data Base?" Personal Computing, (February, 1986):71
- [MA85] Anthony, M. "Merging Spreadsheet Data And Text." Personal Computing, (October, 1985):71-73

- [BSP88] Parker, B. S. "Incorporating Expert System Technology into a Professional Genealogical Information System." Master's Thesis, Kansas State University, 1988
- [RP83] Parker, R. "Computing Your Family Tree." Personal Computing, (January, 1983):112-118
- [JDU82] Ullman, J. D. Principles of Database Systems Rockville, M.A.: Computer Science Press, Inc., 1982
- [EAU85] Unger, E. A. Personal interview. 12, June, 1985
- [DAW86] Waterman, D. A. A Guide To Expert Systems Reading, M.S.: Addison - Wesley Publishing Company, Inc., 1986
- [GW84] Wiederhold, G. "Knowledge and Database Management." IEEE Software (January, 1984): 63-73
- [MW85] Wilson, M. "A Study of Null Values." Masters's Thesis, Kansas State University, 1985

APPENDIX A - BERNSTEIN'S ALGORITHM

INPUT TO ALGORITHM IS A SET OF ATTRIBUTES, A AND A SET OF FUNCTIONAL DEPENDENCIES, F.

STEPS IN ALGORITHM:

1. Eliminate extraneous attributes in each FD of F, producing a new set of FDs, F1.
2. Remove redundant FDs from F1. To do this, find a minimal covering F2 of F1.
3. Partition F2 into groups, where each group has an identical determinant.
4. Find all equivalent keys in F2 by using FD rules.
- 5a. For each pair of equivalent keys $X \rightarrow Y$, $Y \rightarrow X$, remove $X \rightarrow Y$ or $Y \rightarrow X$ or both from F2 if they exist in F2, and add $X \rightarrow Y$ and $Y \rightarrow X$ to H. The new set of FDs, F3 now is the same as F2, with all FDs between equivalent keys removed; H contains the FDs between equivalent keys.
- 5b. Find a minimal covering, F4 of (F3 + H). F4 must include all FDs in H (i.e., all equivalent keys), together with the required FDs in F3 (which may be a subset of F3), to make a minimal covering of F3 + H. Thus $F4 = F3' + H$ where F3' is a subset of F3. The important point here is that F4 must explicitly contain all FDs between equivalent keys.
- 6a. Partition F4 into groups, where each group has an identical determinant.
- 6b. Merge groups with determinants that are equivalent keys.
- 6c. Construct a relation for each group in 6b.

APPENDIX B

FOUR GENERATION ANCESTRAL CHART

NUMBERS IDENTIFY INDIVIDUALS

Benjamin Taylor
40000 King Street
Adamsburg, N.C.
919-555-1212

2(2) Zachary W. TAYLOR
Born 7 May 1880
Place Dubuque Co., Iowa
Married 7 May 1888
Place Dubuque Co., Iowa
Died 1920
Place Marin Co., CA

1(1) Benjamin TAYLOR
Born 7 May 1910
Place Marin Co., CA
Married 7 May 1935
Place Marin Co., CA
Died

Elizabeth ANDREWS
Born 7 May 1910
Place Marin Co., CA
Died

3(3) Mary Fitts ADAMS
Born 7 May 1880
Place Dubuque Co., Iowa
Died 7 May 1950
Place Marin Co., CA

6(6) Benjamin ADAMS
Born 7 May 1848
Place Dubuque Co., IA
Married 317 May 1879
Place Dubuque Co., Iowa
Died 7 May 1890
Place Dubuque Co., Iowa

7(7) Sarah JONES
Born 7 May 1862
Place Radnorshire, Wales
Died 7 May 1930
Place Marin Co., CA
m. (2) WILLIAM HERRY

4(4) [2]
Born Place Died
Place Married Place
Place

5(5)
Born Place Died
Place Married Place
Place

8(8) [4] 2
Born Place Married
Place Died Place

9(9) 3
Born Place Married
Place Died Place

10(10) [5] 4
Born Place Married
Place Died Place

11(11) 5
Born Place Married
Place Died Place

12(12) John ADAMS [2]
Born Place Married 1803
Place Halifax Co., NC [2]
Died 7 May 1828
Place Dubuque Co., Iowa

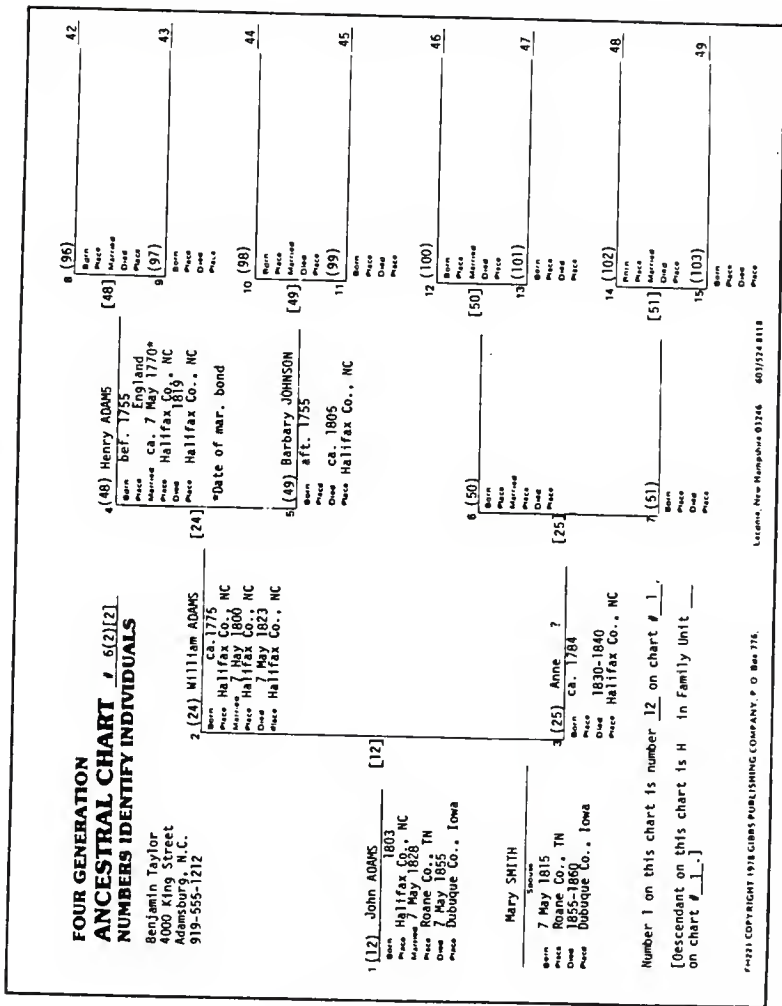
[6] 7 May 1855
Place Dubuque Co., Iowa

13(13) Mary SMITH 7
Born 7 May 1815
Place Roane Co., TN
Died 1855-1860
Place Dubuque Co., Iowa

14(14) [7] 8
Born Place Married
Place Died Place

15(15) 9
Born Place Married
Place Died Place

APPENDIX B



APPENDIX B

NOTE:
 X = Exact
 / = LGS Year
 * = LGS Year
 C = Cent.

FAMILY UNIT CHART

HUSBAND Henry AGANS
 PREPARED BY Benjamin Taylor
 DATE 10 June 1980
 ANCESTRAL CHART # 6-4(48)[24]

WIFE maiden name Barbary JOHNSON
 OCCUPATION "planter"; Sheriff
 STATE or COUNTY England
 COUNTY Halifax
 CITY Halifax
 DATE OF BIRTH 1755
 DATE OF DEATH 1805
 BURIAL CA. 7 May 1770
 BURIAL CA. 1819
 OTHER WIVES
 OTHER HUSBANDS

CHILDREN

SEX	Name	Age	Order of Birth	Birth Date (City, State, Country)	Marriage Date (Name of Spouse)	Birthplace (City, State, Country)	Death Date (City, State, Country)
M	Henry	48.1	1	ca. 1771 (Halifax, NC)	ca. 1795 (Anne Smith)	Halifax, NC	1850-1860 (Halifax Co., NC)
M	John	48.2	2	ca. 1773	7 May 1800 (Mary Johnson)	"	7 May 1833 (Halifax Co., NC)
X	William	(24) 48.3	3	ca. 1775	7 May 1800 (Mary Smith)	"	ca. 1800 (Halifax Co., NC)
F	Maryllis	48.4	4	ca. 1780	John Williams	"	7 May 1815 (Halifax Co., NC)
M	Joseph	48.5	5	ca. 1782	did not marry	"	ca. 1805 (Elizabeth)
M	James	48.6	6	ca. 1785	Elizabeth	"	"
M	Mary	48.6.1	7				
F	Joseph	48.6.2	8				
M	David	48.6.3	9				
			10				
			11				
			12				
			13				
			14				

ANCESTRAL CHART # 6-4(48)[24]
 HALIFAX CO. MARRIAGE BOND
 STATE RECORDS

1980 COPYRIGHT 1978 GIBBS PUBLISHING COMPANY, 51 Church Street, P.O. Box 176, Edmond, N.H. 02246 603-521-8118

APPENDIX C

```

*****
*
*                               P I G . P R G
*-----*
*
*  TYPE:  program library
*
*  CALLED FROM:  This is the starting program
*-----*
*
*  PROGRAMS INVOKED:  INITVAR
*                    INITSCRV
*                    INITSC
*                    MMENU
*-----*
*
*  LOGIC:  This is the initial program executed when
*          entering the FIG 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 initscrV
DO initsc
DO mmenu.prg

```



```

*****
*
*                               M M E N U . P R G
*-----
* TYPE:  program library
*
* CALLED BY:  FIG
*
* 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.*
*
*****
set talk off
SET ECHO OFF
STORE SPACE(1) TO OPT
DO WHILE OPT <> "6"
*
*   drew mester menu on screen
*
  clear
  set 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 sey "SYSTEM"
  @ 9,30 sey "*** Main Menu ***"
  @ 12,28 say "(1) Help Information"
  @ 13,28 say "(2) Add Source Records"
  @ 14,28 say "(3) Delete Source Records"
  @ 15,28 say "(4) Update Source Records"
  @ 16,28 say "(5) Query/Reports  "
  @ 17,28 say "(6) Exit      "
  clear gets
  @ 23,28 say "Enter Selection " get OPT
  read
  do while opt < "1" .or. OPT > "6"
    store " " to OPT
    clear gets
    @ 23,28 sey "Enter Selection " get OPT
    read
  enddo
*
* CALL APPROPRIATE PROCEDURE BASED ON OPTION SELECTED
*
  do case
    case OPT = "1"
      *** Help Information Procedure ***
      do helpinfo
    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 V A R . P R G           *
*-----*
*
* TYPE:  Program Library
*
* CALLED FROM:  FIG, bthcart, mrgcart, dthcart
*
* GLOBAL VARIABLES:  This routine establishes the
*                   global variables used in FIG
*
* 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 files.  Global variables
*         declared here also.
*
*****
PUBLIC dta, city, cnty, st, fac, sax, race, prsno
PUBLIC fnama, mnama, lnama, aga
PUBLIC srcno, src, sfdte, sadta, sfcity, sfcnty, intfd
PUBLIC sfst, sprsno, ral, prsno, srcno
PUBLIC inarr, parson, cnt, opt, birthdats
PUBLIC bdta, bcity, bcnty, bst, caus, bfac, ddta, dcity
PUBLIC dcnty, dst, sname, smnama, sname, mnama, mnama
PUBLIC mlnama, ffname, ffname, flname, baga, bsax, braca
PUBLIC prsno2, prsno3, prsno4, prsno5, prsno6, prsno7
PUBLIC prsno8, f, m, flname, ffname, ffname, mlnama
PUBLIC mfname, mnama, bfname, bnama, blname, fbname
PUBLIC fbmnama, mnama, mbfname, mbmnama, mblname, mdte
**
  use PERSON.dbf
  append BLANK
  stora PERS_NO      to prsno
  store PERS_NO     to prsno2
  store PERS_NO     to prsno3
  store PERS_NO     to prsno4
  stora PERS_NO     to prsno5
  store PERS_NO     to prsno6
  store PERS_NO     to prsno7
  store PERS_NO     to prsno8
*****
  use SOURCE.dbf
  append BLANK
  stora SRC_NO      to srcno
*****
  usa
  return

```

```

*****
*
*           I N I T S C R V . P R G
*-----*
*
*  TYPE:   Program Library
*
*  CALLED FROM:  PIG, bthcert, mrgcert, dthcert
*
*  GLOBAL VARIABLES:  dte, city, cnty, st, fec, sex,
*                    rece, fname, mname, lname,
*                    ddte, dcity, dcnty, dst,
*                    ceus, bdete, bcity, bcnty, bst,
*                    bfec, mdte, ege, bage, beex,
*                    brece, bfname, bname, blname,
*                    ffname, fname, flname, fbname,
*                    mname, mlname, fbfname, fbname,
*                    fbname, mbfname, mbname,
*                    mblname
*
*  LOGIC:  This routine initializes and establishes
*          the type, of the variables associated with
*          BIRTH IN, PERSON, DEATH_IN, and MARRIAGE
*          data 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 bname
store PR_L_NAME to blname
store PR_F_NAME to ffname
store PR_M_NAME to fname
store PR_L_NAME to flname
store PR_F_NAME to fbname
store PR_M_NAME to fbname
store PR_L_NAME to fbname
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
*****
15e
return
```

```

*****
*
*           I N I T S C . P R G
*-----*
*  TYPE:   Program Library
*
*  CALLED FROM:  pig, bthcert, mrgcert, dthcert
*-----*
*
*  GLOBAL VARIABLES:  src, efdte, intfd, sedte,
*                    efcity, sfcnty, sfet
*-----*
*
*  LOGIC:  This routine initielizes end eestablishes
*          the type, of the variablee aesocieted with
*          SOURCE dete bese fields.
*
*****
use SOURCE.dbf
append BLANK
store SRC_TYPE to src
store DATE_SR_FD to efdte
store INT_FD to intfd
store DATE_SR_ET to sedte
store CITY_SR_FD to efcity
store CNTY_SR_FD to sfcnty
store ST_SR_FD to sfet
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
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"
```

```

*****
*
*           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, sfcte, sfcity,
*                   sfnty, 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 intfd
@ 15,2 say "Date entered (MM/DD/YY) " get sedte
@ 15,36 say "Date source found (MM/DD/YY) " get sfcte
@ 16,2 say "Place source was found : "
@ 17,2 say "City " get sfcity
@ 17,29 say "County " get sfnty
@ 17,58 say "State " get sfst

```

M C F M T 1

TYPE: program library

CALLED BY: MRGCERT

GLOBAL VARIABLES: lname, fname, mname, sex, race,
blname, bname, bfname, flname,
ffname, fname, mlname, mfname,
mmname, fblname, fbfname, fbname,
mblname, mbfname, mbmname, bsex,
brace, bage, age

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

@ 4,2 say "Groom Information::"
@ 6,2 say "Last " get ;
lname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 6,28 say "First " get ;
fname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 6,50 say "Middle " get ;
mname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 7,2 say "Sex " get sex picture "|"
@ 7,10 say "Race " get race picture "!"
@ 7,28 say "Age " get age
@ 8,2 say "Groom's Father:"
@ 9,2 say "Last " get ;
flname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 9,28 say "First " get ;
ffname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 9,50 say "Middle " get ;
fmname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 10,2 say "Groom's Mother: "
@ 11,2 say "Last " get ;
mlname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 11,28 say "First " get ;
mfname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 11,50 say "Middle " get ;
mmname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 13,2 say "Bride Information:"
@ 15,2 say "Last " get ;
blname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 15,28 say "First " get ;
bfname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 15,50 say "Middle " get ;
bname picture "!!!!!!!!!!!!!!!!!!!!!"
@ 16,2 say "Sex " get bsex picture "|"
@ 16,10 say "Race " get brace picture "!"
@ 16,28 say "Age " get bage
@ 17,2 say "Bride's Father:"
@ 18,2 say "Last " get ;

```

@ 18,28 say fblname picture "!!!!!!!!!!!!!!!!!!!!!!"
          say "First " get ;
          fbfname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 18,50 say "Middle " get ;
          fbmname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 19,2 say "Bride's Mother (maiden name):"
@ 20,2 say "Last " get ;
          mblname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 20,28 say "First " get ;
          mbfname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 20,50 say "Middle " get ;
          mbmname picture "!!!!!!!!!!!!!!!!!!!!!!"

```

```

*****
*
*                               M C F M T 2
*-----*
* TYPE:  program library
*
* CALLED BY:  MRGCERT
*-----*
*
* GLOBAL VARIABLES:  mdte, fac, city, cnty, st,
*                   wllname, wlfname, wlname,
*                   w2lname, w2fname, s2mname
*-----*
*
* LOGIC:  This is the second screen format of the
*         marriage certificate entry screen routine.
*
*****
@ 2,2 say "Date of Marriage (MM/DD/YY) " ;
      get mdte
@ 4,2 say "Place of Marriage:"
@ 5,8 say "Facility " get ;
      fac picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 6,8 say "City " get ;
      city picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 6,36 say "County " get ;
      cnty picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 6,69 say "State " get st "!!"
@ 8,2 say "Witnesses:"
@ 9,2 say "Last " get ;
      wllname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,28 say "First " get ;
      wlfname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,50 say "Middle " get ;
      wlname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,2 say "Last " get ;
      w2lname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,28 say "First " get ;
      w2fname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 9,50 say "Middle " get ;
      w2mname picture "!!!!!!!!!!!!!!!!!!!!!!"

```

* D C F M T 1 *

* TYPE: program library *
* * *

* CALLED BY: DTHCERT *
* * *

* GLOBAL VARIABLES: lname, fname, mname, ddte, 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. *
* * *

@ 4,2 say "Information on the Deceased:"
@ 6,2 say "Last " get ;
@ 6,28 say "First " get ;
@ 6,50 say "Middle " get ;
@ 8,2 say "Date of Death (MM/DD/YY) " get ddte
@ 9,2 say "Place of Death City " get ;
@ 9,43 say " County " get ;
@ 9,71 say " State " get dst picture "!!!"
@ 10,2 say "Sex " get sex picture "!"
@ 10,10 say "Race " get race picture "!!!"
@ 10,28 say "Cause of Death " get ;
@ 12,2 say "Date of Burial " get bdte
@ 13,2 say "Burial Site: City " get ;
@ 13,41 say "County " get ;
@ 13,69 say "State " get bst picture "!!!"
@ 14,2 say "Burial Facility " get bfac picture "!!!"
@ 16,2 say "Date of Birth " get dte
@ 17,2 say "Place of Birth: city " get ;
@ 17,44 say "County " get ;
@ 17,72 say "State " get st picture "!!!"

```

*****
*
*           D C F M T 2
*-----*
* TYPE:   program library
*
* CALLED BY:  DTHCERT
*-----*
*
* GLOBAL VARIABLES:  lneme, fname, mneme, ddte, dcity,
*                   dcnty, dst, sex, rece, ceus,
*                   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  sey "Deceased:"
@ 2,3  sey "Last ", ;
      lneme picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 2,28 sey "First ", fname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 2,50 sey "Middle ", mname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 4,2  sey "Spouse: "
@ 5,2  say "Last " get ;
      slname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 5,28 sey "First " get ;
      sfname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 5,50 sey "Middle " get ;
      smname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 7,2  sey "Fether: "
@ 8,2  say "Last " get ;
      flname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 8,28 say "First " get ;
      ffname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 8,50 sey "Middle " get ;
      fmneme picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 10,2 say "Mother: "
@ 11,2 say "Last " get ;
      mlneme picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 11,28 say "First " get ;
      mfname picture "!!!!!!!!!!!!!!!!!!!!!!"
@ 11,50 sey "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 certificete screen(s) is printed. *
*           The user is prompted to enter the death *
*           certificete deta. The deta is validated. *
*           If errors are found, the user is prompted *
*           to reenter the fields in error. If no *
*           errors, the source deta is added to the *
*           appropriate deta 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 Deta";
get opt picture "!"
read
if opt = "X"
return
endif
do veldc
do velsrc
if inerr = "y"
@ 22,2 say "Reenter the field(s) in error"
endif
loopcnt1 = inerr

```



```

@ 1,1 clear
enddo
*
* Is this e 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

  case cnt = 1 .end. lname <> " "
    do crtsrcno
    do crtprsnno
    do eddprsn
    do eddsrcc
    do adddc
    if birthdetc = "Y"
      do addbc
    endif

  ** add deceased spouse

  case cnt = 2 .end. slname <> " "
    do crtprsnno
    do eddsrcc
    do addmc
    fneme = sfname
    mneme = smname
    lneme = slname
    do addprsn
    if sex = "F"
      rel = "HUSBAND"
    else
      rel = "WIFE "
    endif
    do addrel

  ** add deceased fethar

  case cnt = 3 .and. flname <> " "
    do crtprsnno
    do addsrcc
    fneme = ffname
    mneme = fmname
    lneme = flname
    do eddprsn
    rel = "FATHER"
    do eddrel

  ** add deceased's mother

  case cnt = 4 .end. mlname <> " "
    do crtprsnno

```

```
do addsrc
  fname = mfname
  mname = mmname
  lname = mlname
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, inarr, rel, opt,
*                    birthdate, parson
*-----*
*
*  LOGIC:  Tha birth cartificate screan(s) is printed.
*          The user is prompted to antar tha birth
*          cartificate data. Tha data is validatad.
*          If errors are found, the user is prompted
*          to raentar the fialde in arror. If no
*          arros, tha sourca data is addad to the
*          appropiata data basa filas.
*
*****
clear
src   = "BC"
cnt   = 1
loopcnt1 = "y"
opt   = " "

@ 1,32 say "ENTER BIRTH CERTIFICATE"

do while cnt < 4
@ 3,2 clear
do whila loopcnt1 = "y"
do case
case cnt = 1
@ 3,2 say ;
"Infant Birth Certificate Information:"
case cnt = 2
@ 3,2 say ;
"Father Birth Certificata Information:"
case cnt = 3
@ 3,2 say ;
"Mochar Birth Cartificate Information:"
andcasa
do bcfmt
if cnt = 1
do srcfmt
endif
@ 21,2 say ;
"Typa X to Exit, any other key to antar data";
gat opt
raad
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 birthdate = "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
use
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,
*                     CRTPRSNO, ADDSRC, ADDMC,
*                     ADDPRS, INITSC, INITSCRV
*                     INITSC, INITSCRV
*
*   GLOBAL VARIABLES:  cnt, inarr, ral, opt, eax, bsex,
*                     fname, mname, lnama, bfname,
*                     bmname, blname, mfnama, mmnama,
*                     mlname, ffnama, fmnama, flname,
*                     fbfname, fbmnama, fblname,
*                     mbfnama, mbmnama, mblname,
*                     wlfname, wlmnama, wllnama,
*                     w2fname, w2mnama, w2lnama
*
-----
*
*   LOGIC:  Tha marriaga cartificata screen is printed.
*           The user is prompted to enter the marriage
*           certificate data.  The data is validatad.
*           If arrore are found, the usar is prompted
*           to reantar tha fields in arror.  If no
*           errors, tha sourca data is addad to tha
*           appropriata data base files.
*
*****
claar
src   = "MC"
cnt   = 1
loopcnt1 = "Y"
opt   = " "

@ 1,30 say "ENTER MARRIAGE DEATH CERTIFICATE"

do while loopcnt1 = "Y"
do mcfmt1
@ 20,1 say " "
wait
@ 1,1 claar
do mcfmt2
do srcfmt
@ 21,2 eay ;
      "Type X to Exit, any other key to Enter Data";
      gat opt pictura "!"
read
if opt = "X"
      return
endif
do valmc
do valsrc

```

```

if inerr = "Y"
  @ 22,2 sey "Reenter the field(s) in error"
endif
loopcnt1 = inerr
@ 1,1 cleer
enddo
*
* Is this e new person, or does this person exist?
*
* >>>>>> code goes here
*
* Enter death certificete 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 edded.
*
cnt = 1
do while cnt < 9
do case

  ** Add groom

  case cnt = 1 .end. lname <> "
    do crtsrcno
    do crtprsn0
    do addprsn
    do eddsrc
    do eddmc
    rel = "SELF"
    do eddrel

  ** Add bride

  case cnt = 2 .end. blname <> "
    do crtprsn0
    do addsrc
    fname = bfname
    mname = bmneme
    lname = blname
    eddprsn
    do eddmc
    rel = "WIFE"
    do eddrel

  ** Add groom's fether

  case cnt = 3 .end. flname <> "
    do crtprsn0
    do eddsrc
    fname = ffname
    mname = fmneme
    lname = flname
    addprsn
    do eddmc
    rel = "FATHER"
    do eddrel

  ** Add groom's mother

  case cnt = 4 .end. mlname <> "

```

```

do crtprsno
do eddsrsc
fname = mfname
mname = mmname
lneme = mlname
eddsprs
do eddmc
rel = "MOTHER"
do eddrel

** Add Bride's fether

cese cnt = 5 .end. fblname = "
pr sno = prsno2
do crtprsno
do eddsrsc
fname = fbfname
mname = fbmname
lneme = fblname
eddsprs
do eddmc
rel = "FATHER"
do eddrel

** Add Bride's mother

cese cnt = 6 .end. mblname = "
pr sno = prsno2
do crtprsno
do eddsrsc
fname = mbfname
mname = mbmname
lneme = mblname
eddsprs
do eddmc
rel = "MOTHER"
do eddrel

** Add Witness 1

cese cnt = 7 .end. wllname = "
do crtprsno
do eddsrsc
fname = wlfname
mname = wlmname
lneme = wllname
eddsprs
do eddmc
rel = "WITNESS"
do eddrel

** Add Witness 2

cese cnt = 8 .end. w2lname = "
do crtprsno
do eddsrsc
fname = w2fname
mname = w2mname
lneme = w2lname
eddsprs
do eddmc
rel = "WIFE"
do eddrel
endcese
cnt = cnt + 1
enddo
use
do initsc
do initscr
return

```

```

*****
*
*                               D E L T S R C . P R G
*
*-----*
*   TYPE:  Commend 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
etore " " to erctyp
do while srctyp <> "5"
  clear
  set bell off
  store space(1) to erctyp
  @ 3,19 to 6,60
  @ 7,19 to 19,60
  @ 5,30 sey "SOURCE RECORD ENTRY"
  @ 9,26 sey "Primary Source Record Types:"
  @ 12,26 sey "(1) Birth Certificete"
  @ 13,26 sey "(2) Merriage Certificete"
  @ 14,26 sey "(3) Deeth Certificete"
  @ 15,26 sey "(4) Informel Source"
  @ 16,26 sey "(5) Exit - return to main menu"
  clear gets
  @ 21,28 sey ;
  "Enter Source Record Type: " get srctyp
  read
  do while erctyp < "1" .or. erctyp > "5"
    store " " to erctyp
    clear gets
    @ 23,28 sey ;
    "Enter Source Record Type: " get erctyp
    read
  enddo
*
* cell appropriate procedure beeed on source
* type selected
*
do case
  case erctyp = "1"
    do dbthcert
  case erctyp = "2"
    do dmrgcert
  case erctyp = "3"
    do ddthcert
  case erctyp = "4"
    do dinfcert
  case erctyp = "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 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

```

```

*****
*
*                               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 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 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
```

```

*****
*
*                                OBTHCERT.PRG
*
*-----*
* CALLED FROM:  DELT SRC.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, OELTPERS, OELT SRCE,
*                  OELTEC, OELTREL, INITSCRV, INITSC
*
*-----*
*
* ACTIVE DATABASE FILES:  BIRTH_IN.OBF, SOURCE.OBF,
*                        PERSON.OBF
*
* ACTIVE INDEX FILES:    BIRTH_IN -> SRCPERS.NDX
*                        SOURCE  -> PERSTYPE.NDX, SPERS.NOX
*                        PERSON  -> NAME.NOX, NAME2.NOX,
*                        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 *
*
```

```
loopcntl = "y"
do while loopcntl = "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 "IAST " 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
enddo while loopcntl
```

```
*
* 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 BC *
*
```

```
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 BIRTH_DATE->BIRTH_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 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 INITSRV
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, DELTSCRC,                 *
*                   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 = " "
pho = " "
match = "y"
styp = "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 FRDM:  DELTSRC.PRG                                           *
*
* LOGIC:      THIS IS THE PROGRAM THAT HANDLES THE LOGIC          *
*              REQUIRED TD DELETE A DEATH CERTIFICATE.              *
*              THE USER SUPPLIES THE NAME FOR THE PRIMARY          *
*              PERSON ON THE DC THEN THE PROGRAM VALIDATES        *
*              THE NAME AND SEARCHES FDR THE APPRDPRIATE          *
*              NAME COMBINATION.  THE PROGRAM PROVIDES A          *
*              LIST OF ALL NAMES AND THEIR ASSOCIATED              *
*              PERSON NUMBERS AND SDCURCE NUMBERS THAT            *
*              MATCH WHAT THE USER SPECIFIED.  THE USER          *
*              THEN SELECTS THE APPROPRIATE PERSDN AND            *
*              SOURCE.  THIS PROGRAM THEN INVOKES THE              *
*              PROGRAMS REQUIRED TD DELETE DATA FRDM ALL           *
*              DATABASE FILES REQUIRED.                               *
*-----*
*
* PROGRAMS INVOKED: VALNAME.PRG, DELTPERS, DELTSRCE,                *
*                  DELTDC, DELTREL, INITSCRV, INITSC                *
*-----*
*
* ACTIVE DATABASE FILES:  DEATH IN.DBF, SOURCE.DBF,                 *
*                        PERSON.DBF                                  *
*
* ACTIVE INDEX FILES:    DEATH_IN -> DSRCPPERS.NDX                  *
*                        SOURCE   -> PERSTYPE.NDX, SPERS.NDX        *
*                        PERSDN   -> NAME.NDX, NAME2.NDX,           *
*                        NAME3.NDX                                  *
*****
*
* INITIALIZE VARIABLES *
*
clear
opt = " "
pno = " "
match = "y"
stype = "DC"

*
* ESTABLISH ACTIVE WDRK AREAS *
* WITH APPRDPRIATE DATABASE FILES *
* AND THE REQUIRED INDEX FILES *
*
* ALSD, ESTABLISH RELATIONS *
*
select 3
use DEATH_IN index DSRCPPERS alias DEATH
select 2
use SDURCE 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
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 DC *
*
```

```
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 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
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,
*                DTHCERT.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 INFORMTION    *
*               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, sfdate, sfcity,
*                     sfcnty, 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 intf
replace DATE_SR_ET with sedte
use
```

```

*****
*                                     *
*                               ADDREL *
*-----*
*
* INVOKED BY:  BTHCERT.PRG, MRGCERT.PRG, *
*              DTHCERT.PRG              *
*
* LOGIC:      THIS PROGRAM PROVIDES THE *
*              ACTUAL ADDITION OF RELATIONSIPS *
*              INTO THE RELATION DATABASE. *
*-----*
*
* 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 PER_S_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

```

```

*****
*                                     *
*                                     ADDBC                                     *
*-----*
* 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, *
*                   dte, 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 dtc  
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
```

```

*****
*                                     *
*                                     *
*-----*
*                                     *
* 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 dte
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
```

DESIGN OF A PROFESSIONAL
GENEALOGICAL INFORMATION SYSTEM:
INCLUDING NAVIGATION FROM AN
UNSTRUCTURED DATABASE TO A
STRUCTURED DATABASE

by

ELLEN J. BAILEY

B.S., East Tennessee State University, 1980

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

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 for the professional genealogist. The thesis includes a discussion of a the genealogical process, the design and implementation of a relational database to support the information system.

The information systems has been designed as four subsystems, the user front end, the rule-based expert system, the data base control system and the text manipulation systems. The implementation system was d-Base III+.