A GENERALIZED NAME INDEXING METHOD FOR IMAGE DATABASES

Robert B. Garvey

Robert L. Womack IV

Witan Inc.
Kansas City, Missouri

A Generalized Name Indexing Method for Image Databases

Robert B. Garvey
Witan, Inc.
Robert L. Womack, IV
Third Judicial District Court
of Kansas

March 13, 1981

1. Name Searching -- The Problem Part 1

Traditional approaches to searching a list of names usually involve a KSAMish approach to the data file: i.e. positioning the program (or a terminal user) at the beginning of a group of names spelled in like manner, and then performing additional subsetting or processing of the group of names. KSAM (or ISAM) is typically used for such applications because of their ability to position the record pointer using only a partial key. IMAGE, on the other hand, is seldom used for such applications since inquiries must be made against a complete search item. For applications in which inquiry against an existing "people" base is required, the complete entry of a name may be at best redundant and at worst extremely subject to error since variations in spelling of a name at data entry time from that of the master name on file are difficult to handle. Criteria for a first cut name search algorithm should then include at least the following:

- .The ability to search on a partial or incomplete name
- .The ability to operate upon or display to a terminal user a group of similar names.

Name Searching -- Problem Part 2

An additional requirement for many name search applications is the ability to search for names by phonetic equivalent, that is names should be selected for analysis based upon how they sound rather than how they are spelled. Applications which must employ field originated forms are typically in need of such capability. A second-order name search technique would, therefore, require the additional capability of inquiry by the phonetic value of a name rather than (or perhaps, in addition to) the exact spelling or partial spelling of a name. Lastly, in order to increase the "crash-worthiness" and ease of maintenance of a name search system, the person data base should be a part of an organization's IMAGE data base. The ideal name search technique then should have the following attributes:

.The ability to search on a partial or incomplete name
.The ability to operate upon or display to a terminal user a group of similar names.
.The ability to search by a phonetic key
.Suitability for implementation using IMAGE

The remainder of this paper will etch out a partial data base, compare two phonetic encryption methods, and suggest an approach to a name inquiry system which will satisfy the above requirements.

2. Phonetic Name Searching and IMAGE

Phonetic name encryption gives the system designer a tool which enables the design of name inquiry systems that meet the four criteria set out in part one of this paper. In this section a partial data base schema to support name indexing will be outlined and standards for evaluating name encryption algorithms will be suggested and applied. The development of synonyms is at the heart of any phonetic name encryption module. It has been suggested that two standards should be used to evaluate phonetic name encryption algorithms: reliability and selectivity (1). In discussing these concepts in a study prepared for the New York State Identification and Intelligence System, Robert Taft defines reliability as , "the probability that the technique will retrieve the correct suspect it, in fact, the suspect is in the file." (2) Continuing the analysis, Taft states selectivity is,

> "the average percentage of [a] file that the technique will accept on an average search request. If, for example, a method has a selectivity factor of one percent, then the method will, on the average, accept one percent of the file and reject ninety nine percent ... the selectivity factor is an accurate measure of the amount of work the system will have to perform." (3)

To these criteria this writer will add yet a third,: The algorithm should be fast in execution time and require a minimal code and data segment on the HP 3000. A simple IMAGE data structure implied by the mechanics of phonetic name transformation techniques is shown below in Figure 1.

Figure 1

```
 ..............
 . Phonetic .
  .  Code   .                          ...................
   .(Auto).------------------------->.   Name Detail   .
    .    .   Primary Key: Phonetic     .               .
     .  .                 Code          .             .
      ..                                 .           .
                                          ..........
```

Since the object of the encryption algorithm is to group similar sounding names into the same group (i.e. assign them the same search item value), a straight-forward representation of this in a data base is a detail set with a primary key of the phonetic name code indexed by an automatic master. The detail set allows for multiple records with the same search item value; making the phonetic key the primary key of the detail set will speed access by insuring that records with the same phonetic key will be stored contiguously on disk after a DBUNLOAD/DBLOAD cycle. This same set may be indexed by other master sets and should contain at a minimum the name and name associated information for the entity being indexed.

SOUNDEX and NYSIIS Name Encryption Techniques

In this section two name encryption techniques will be specified and evaluated using the standards outlined in Section 2. The SOUNDEX algorithm is an especially common name grouping technique in use today. Its implementation is extremely straight-forward on the HP 3000 (see appendix A). The algorithm upon which the SPL procedure displayed in Appendix A is based is given below in Table 1.

Table 1 - SOUNDEX Variant 1 Alogrithm (4)

	Rule	Example
		Original name ASHCROFT
1.	Remove the letters "W" and "H" from the name	ASCROFT
2.	Code all letters using the following table and using "0" for vowels:	0226013

B,F,P,V	= 1
C,G,J,K,Q,S,X,Z	= 2
D,T	= 3
L	= 4
M,N	= 5
R	= 6

3.	Make all multiple digits single	026013
4.	Remove all zeros after the first position of the value	02613
5.	Add sufficient zeros on the right hand side to make six digits	026130
6.	Replace the first digit with the first character of the name	A26130

Taft evaluates the reliability of this technique at 95.99 percent. Its selectivity factor is .213 percent, i.e., this method will on the average retrieve .213 percent of a file as matches to a name inquiry (5). The procedure shown in Appendix A takes 69 CPU seconds to encode 5,524 names on an HP 3000 Series III. The NYSIIS name encryption technique represents an effort to develop a phonetic code that is both more selective than SOUNDEX procedures and more effective at handling the "orthographic errors which occur in the recording of 'Spanish' and other South European names" (6). The NYSIIS algorithm is outlined in Table 3, below:

Table 3 -- The NYSIIS Algorithm (7)

Rule

1. If the first letters of the name are
   "MAC" then change these letters to "MCC"
   "KN" then change these letters to "NN"
   "K" then change these letters to "C"

```
          "PH"  then change these letters to "FF"
          "PF" then change these letters to "FF"
          "SCH" then change these letters to "SSS"
2.     If the last letters of the name are
          "EE" then change these letters to "Y "
          "IE" then change these letters to "Y "
          "DT","RT","RD","NT","ND" then
             Change these letters to "D "
3.     The first character of the NYSIIS code is the
       first character of the name
4.     In the following rules, a scan is performed on
       the characters of the name.  This is described
       in terms of a program loop.  A pointer is used
       to point to the current position under
       consideration in the name.
       Step 4 is to set the pointer to the second
       character of the name.
5.     Considering the position of the pointer, only
       one of the following statements can be
       executed:
       If blank then go to rule 7
       If the current position is a vowel (AEIOU)
          then
          If equal to "EV" then change to "AF"
          otherwise change current position to "A"
       If the current position is the letter
          "Q" then change the letter to "G"
          "Z" then change the letter to "S"
          "M" then change the letter to "N"
       If the current position is the letter "K" then
          If the next letter is "N" then
             replace the current position by "N"
          otherwise
             replace the current position by "C"
       If the current position points to the letter
          string "SCH" then
           replace the string with "SSS"
       otherwise If the current position points to
          the letter string "PH" then
          replace the string with "FF"
       If the current position is the letter "H" and
          either the proceding or following letter is
          not a vowel (AEIOU) then replace the current
          position with the preceding letter
       If the current position is the letter "W" and
          the preceding letter is a vowel then
             replace the current position with the
                preceding position
       If none of these rules applies, then retain
          the current position letter value
6.   If the current position letter is equal to the
     last letter placed in the code then
        set the pointer to point to the next letter
        go to rule 5
```

```
7.  If the last character of the NYSIIS code is
       the letter "S" then
       remove it
8.  If the last two characters of the NYSIIS code
       are the letters "AY" then
       replace them with the single letter "Y"
9.  If the last character of the NYSIIS code is the
       letter "A" then
       remove this letter
```

Taft evaluates the reliability of this technique at 98.72%. The selectivity factor of the NYSIIS routine is .164% (7). A program using an SPL implementation of the NYSIIS algorithm developed by the Illinois Law Enforcement Commission (8) took 66 CPU seconds to encrypt 5424 names.

Detailed Comparison

The SOUNDEX and NYSIIS algorithms were compared using the following procedure:

1. A file of 5424 names was encrypted using each technique
   A sequential MPE file containing the phonetic code for each name was generated.
2. Timings were derived for both processes
3. The files of phonetic codes were sorted in ascending order
4. Duplicate codes in each file were eliminated.
   Each code was tagged with the number synonyms for that code that had been generated.
5. Summary statistics were derived using SPSS (9).

The results of the SPSS runs and other summary measures are given in Table 4, below:

Table 4 -- SOUNDEX vs. NYSIIS

Routine Name	CPU Time	Segment Size	# Codes Generated	Avg. No. Synonyms/ Code	STD DEV	Maximum No. Synonyms/ Code
NYSIIS	66	1021	1673	2.895	4.644	77
SOUNDEX	69	434	1478	3.669	5.992	84

The comparison data displayed in Table 4 reveals that the performance characteristics of the NYSIIS routine are on the average about twenty percent more efficient, both in terms of the number of entries generated and the average number of synonyms per entry, than SOUNDEX. It should be noted, however, that applications using one algorithm as opposed to the other will not necessarily show equivalent differences in performance. This is especially true for applications using IMAGE databases which are perodically DBUNLOAD/DBLOADed. Empirical data at the Third District Court indicate that IMAGE will block moderate sized name detail sets (for example, doubly keyed records with 50 byte name field, six byte soundex primary key, ten byte secondary key and eight bytes of miscellaneous data) at from ten to eleven blocks per physical record. Thus one disc I/O will all matching phonetic entries all but five to seven percent of the time, depending upon the algorithm selected. If DBUNLOAD/DBLOADs are not done on a periodic basis, then performance differentials should approach that found in the sample data used in this paper.

Summary

Applications using either of the phonetic encryptions algorithms discussed in this paper will be able to search an appropriatly keyed detail set by phonetic key. Both methods generate fixed length keys. Finally, by doing phonetic searches on the last name and exact or weighted matches on the remaining characters in a name specified at data entry time, an application program may both search on an incomplete name and develop a subset of matching names for further analysis.

ENDNOTES

(1) Robert L. Taft, "Name Search Techniques," Albany: New York State Identification and Intelligence System, undated, pp 37-38. This paper is an excellent analysis of the merits and demerits of several different name encryption systems.

(2) *Ibid.*, p. 37.

(3) *Ibid.*, p. 38.

(4) *Ibid.*, p. 58.

(5) *Ibid.*

(6) *Ibid.*, p. 86.

(7) *Ibid.*, pp. 88-90.

(8) NYSIIS was implemented on the HP 3000 by J. David Coldren and his staff at the Illinois Law Enforcement Commission in May, 1977.

(9) See Norman H. Nie, *et. al.*, *Statistical Package for the Social Sciences, 2nd Edition.*, New York: Mc Graw Hill and Company, pp 194-202 for an explanation of the FREQUENCIES procedure used to analyze the data referenced in this paper.

Appendix A

```
$CONTROL SUBPROGRAM,LIST,SEGMENT=RUSSELLSOUNDEX
BEGIN
PROCEDURE SOUNDEX(NAMESTRING,SOUNDEX,ERRCODE);
LOGICAL ARRAY NAMESTRING,SOUNDEX;
INTEGER ERRCODE;
BEGIN
   BYTE ARRAY NAMESTRING'B(*)=NAMESTRING;
   BYTE ARRAY SOUNDEX'B(*)=SOUNDEX;
   BYTE ARRAY NAMESTRING'S(0:50),STR(0:50);
   BYTE ARRAY RUSSELL'SOUNDEX(0:255);
   INTEGER I,J;
   INTEGER STRING'LENGTH;
   BYTE FIRST'LETTER;
   INTRINSIC CTRANSLATE;

   MOVE RUSSELL'SOUNDEX:=65("0"),2;
      MOVE *:="01230120022455012623010202000000012301200224550",2;
      MOVE *:="12623010202",2;
      MOVE *:=133("0");
   MOVE NAMESTRING'S:=NAMESTRING'B,(50); << COPY NAME TO WORK AREA >>
   NAMESTRING'S(50):=" ";      << SET TERMINATOR BYTE >>
   SCAN NAMESTRING'S UNTIL ", ",1;      << SCAN FOR END OF LAST NAME >>
   STRING'LENGTH:=TOS-@NAMESTRING'S+1;  << SET STRINGLENGTH >>
   MOVE STR:=NAMESTRING'S,(STRING'LENGTH); << MOVE LASTNAME INTO STR >>
   FIRST'LETTER:=STR;                << SAVE FIRST LETTER OF LAST NAME >>
   IF INTEGER(FIRST'LETTER)>90 THEN
      FIRST'LETTER:=BYTE(INTEGER(FIRST'LETTER)-32);
   IF FIRST'LETTER <> ALPHA THEN
      BEGIN
      ERRCODE:=1;
      RETURN;
      END;
   CTRANSLATE(0,STR,STR,STRING'LENGTH,RUSSELL'SOUNDEX);
   IF  <  THEN BEGIN                     << TEST FOR SUCCESSFUL >>
      ERRCODE:=1;                        << TRANSLATION AND SET >>
      RETURN;                            << ERRCODE ACCORDINGLY >>
      END;
   ERRCODE:=0;                           << INITIALIZE ERRCODE >>
   FOR I:=0 STEP 1 UNTIL STRING'LENGTH-2 DO
      BEGIN
      J:=1;
      WHILE STR(I)=STR(J+1) AND J<=STRING'LENGTH-1 DO
         BEGIN
         STR(J+1):="0";
         J:=J+1;
         END;
      END;
   J:=1;
   FOR I:=1 STEP 1 UNTIL STRING'LENGTH-1 DO  << SQUEEZE OUT ZEROS >>
      IF STR(I)<>"0" AND J<6 THEN
         BEGIN
         SOUNDEX'B(J):=STR(I);
         J:=J+1;
         END;
   IF J<6 THEN                           << POSTFIX ZEROS TO MAKE >>
                                         << SIX DIGIT CODE       >>
      FOR I:=J STEP 1 UNTIL 5 DO
         SOUNDEX'B(I):="0";
   SOUNDEX'B:=FIRST'LETTER;              << PREFIX SOUNDEX CODE WITH >>
```

```
                                            << FIRST LETTER OF LAST      >>
                                            << NAME                      >>
      END;
   END.
```