Is Soundex Good Enough for You?
On the Hidden Risks of Soundex-Based Name Searching

Frankie Patman and Leonard Shaefer

2214 Rock Hill Road, Suite 201
Herndon, VA 20170
www.las-inc.com

Contents

I. Introduction

Despite many remarkable advances made recently in the automated processing of natural languages, automated processing and matching of names in databases or free text has languished for many decades without significant theoretical or practical advances.

The problem to be solved is a familiar one for many people: a customer is entered in one database with the surname "Rodgers," and in a different database as "Rogers." A client's name is recorded as "Dayton," but should actually be spelled "Deighton." A Chinese family has one set of information recorded under the surname "Xiao," and another under the surname "Hsiao."

Globalization of trade and the ever-growing ethnic diversity of the American populace combine to transform what were once marginal cultural issues into crucial challenges for many software engineers and IT professionals. While it may be possible in some instances simply to work around problems of name complexity by relying on other types of information, valuable information and competitive advantages are often lost in the process. This loss grows still more evident when the trend towards more personalized communications and customer relationship management are considered. Getting names right is a key skill for any organization, no matter what its mission may be. And never has the challenge of dealing with variations in name spelling been greater than it is today.

Efforts to deal with the complexity of names precede the rise of data processing by a good half-century. The earliest attempt at coping with name variation was the Soundex

matching algorithm, developed in the early years of the 20th century as an aid for manual filing of U.S. Census records. The original Soundex method (as well as its many variants) was implemented as a software-based algorithm, and is today perhaps the most widely used alternative to exact-matching when names are involved in automated search and retrieval systems.

Soundex is indeed a hardy and long-lived technique, and has much to recommend it: it is non-proprietary, relatively fast, efficient and generally effective for certain well-known types of spelling variation associated with many commonly encountered names. Best of all, Soundex is free. It comes as a built-in function in many DBMS products, programming languages and data management tools. No surprise, then, that it is the tool of choice for many application developers who must address the need to match, search and retrieve names.

However, Soundex proves in practice to be limited in dealing with many kinds of variation inevitably present in collections of names. This paper will point out and exemplify several key areas where the Soundex name-matching algorithm performs poorly. Subsequently, an alternative approach to the automated name-matching problem will be described.

The goal of this paper is to clarify the hidden risks that application and database developers assume when relying primarily on Soundex as a means for matching and retrieval of names.

II. Description of Soundex

The term "Soundex" actually covers several variations of an algorithm first developed and patented by Robert C. Russell in 1918. Most versions of Soundex convert a surname into a code consisting of the first (leftmost) letter of the surname, followed by three (or more in some cases) digits. The digits are assigned according to a pre-determined grouping of consonants, where the consonant groups share phonetic features (that is, sound similar in one or more ways). This is the key concept behind Soundex: a constant relationship between letters and sound should assure that similar-sounding names are assigned the same code.

The standard Soundex algorithm defines the following groups:

Letter	***Code***
B,F,P,V	1
C,G,J,K,Q,S,X,Z	2
D,T	3
L	4
M,N	5
R	6
H,Y,W	(omitted)
A,E,I,O,U	(omitted)

In this canonical algorithm, the leftmost letter is always retained, and all non-initial vowels as well as non-initial H, Y, and W are omitted. Only one digit is used for consecutive letters that result in the same code (*e.g.*, CK = 2, not 22); codes with more than three digits are truncated to the leftmost three digits; and codes for names with fewer than three consonants are padded with zeros (*e.g.*, PEEL = P400). The large number of characters in category 2 results from overlapping relationships between consonants. The letter "C," for example, is related to both "S" (which is in turn related to "Z") and "K" (which is related to "G" and "Q").

Soundex allows names with similar pronunciations but disparate spellings to be retrieved from a single query. For example, PATER will also retrieve PAIDER (both have Soundex Code P360), SOMERS will retrieve SUMMERS and SOMMARS (all with Code S562), and GARDNER will match GARDINER and GARTNER (Code G635).

Variations of the original Soundex method were later introduced as limitations became apparent. These include techniques such as breaking the consonant groups into more closely related sets (*e.g.*, {B,P}, {F,V}, {C,K,S}, {G,J}, {Q,X,Z}, {D,T}, {L}, {M,N}, {R}); allowing for more than three digit places; coding certain consonant clusters that represent a single sound as a single digit (*e.g.*, TCH, DG); creating multiple codes for consonant clusters with multiple pronunciations (*e.g.*, CH, as in 'Christian' and 'Charlotte'); and coding initial letters just like other letters (*e.g.*, initial C and K would have the same code, so that 'Kerr' and 'Carr' would match).

All versions of Soundex attempt to capture phonetic similarities without taking into account the surrounding context in which a letter occurs, so that a numeric value can be assigned to individual consonants regardless of letters that precede or follow it. Later attempts to offset this lack of contextual information in the original Soundex algorithm include the Phonex and the Daitch-Mokotoff Soundex system, both of which make limited use of contextual information in determining which numeric codes to assign to a name.

A non-phonetic compression algorithm was developed by Leon Davidson in the early 1960's for use in airline passenger tracking systems. Davidson's algorithm simply drops all vowels, as well as double consonants and the letters H, Y, and W. It does not group consonants in any way. Studies of this method have not shown it to be more effective than Soundex in general (Hermansen 1985).

The computational benefits of Soundex-type algorithms are easy to see: by exchanging a name for a code, all variant spellings of the name can be expected to share that same code, allowing a relatively efficient search of a small subset of a database, versus "brute force" evaluation of every name as a potential match. Soundex keys are typically used to form an index for data implemented in relations DBMS products, allowing very fast key-based retrievals of a (theoretically) small number of potentially matching names.

III. Evaluations of Soundex-type algorithms

While it is certainly compact and efficient, Soundex-type approaches still fall well short of solving many of the problems associated with searching for names. Two recent studies looked at the performance of the basic Soundex algorithm, using statistical measures to gauge accuracy. Alan Stanier (September 1990, *Computers in Genealogy*, Vol. 3, No. 7) extracted all 411,716 surnames from the 1851 U.S. Census sample and linked related name forms based on information provided in a dictionary of surnames. Calculating search results for each name in the sample, he found that only thirty-three percent of the matches that would be returned by Soundex would be correct. Even more significant was his finding that fully twenty-five percent of correct matches would fail to be discovered by Soundex.

A second study by A. J. Lait and B. Randell (1996) compared the performance of several name-matching algorithms, including the basic Soundex method. Searches were conducted on a data set of 5600 unique surnames, chosen to represent names beginning with each letter of the alphabet at a frequency of occurrence reflecting actual alphabetic distributions of names, and including as well names of varying lengths. The study found that Soundex (judged to be the best of the four algorithms compared) returned only 36.37% of the actual correct matches, and that more than sixty percent of names that were correct matches for query names were not returned.

Studies such as these raise serious questions about whether it is appropriate to use Soundex as the basis for name-searches in applications where accuracy, completeness and efficiency are crucial. For a significant percentage of search transactions, there is a considerable risk that critical information may lie buried within a database, undiscovered by searches depending on Soundex. Soundex-based name search systems place an undue burden on the user, who must try various strategies in order to offset the rigid constraints of a fixed letter-to-sound relationship that lies at the very heart of key-based name retrievals.

While it is certainly true that a key-based search relying on Soundex can render results quickly, this efficiency is largely lost if each transaction must be repeated many times by the user, in order to produce acceptable results.

The studies cited above make the general case that one cannot rely solely on Soundex if the goals of a name search include finding as many of the related names as possible, or if the time needed to dig through potentially large numbers of irrelevant returns to find the desired records is not available**.** They say very little, however, about specific factors that constrain Soundex's accuracy and efficiency.

In the next section, we will present eleven problems that can result in crucial connections between names being missed or obscured. Some of these focus on design limitations within the Soundex method, and others relate to the content or structure of names in databases or free text.

IV. What Soundex cannot do

Each of the issues discussed below is illustrated with actual returns from Soundex searches in sample databases. The example queries and returns are found in the appendix.

1. ***Dependence on initial letter***. The Soundex algorithm uses the first letter of a name as a key component of the code it generates to represent the name. Names that do not begin with the same letter will never match each other. A data entry operator hearing the name "Korbin" might type in the much more common "Corbin." Although "Korbin" may be in the database, it will not be returned by a standard Soundex search on "Corbin." (Appendix, item 1.)

2. ***Noise intolerance***. Because names are resistant to standard data-validation and data-quality techniques, random keying errors are unavoidable in collections beyond trivial size. If a database record contains the name "Msith" ("Smith" with a common transposition keying error), Soundex cannot overcome this simple transposition when searching for the correctly spelled "Smith." If "Hubbins" was really meant to be "Huggins," Soundex will be of no use. In general, Soundex relies on predictable sound-to-letter relationships, so it will not overcome any random spelling variations, unless these just happen to coincide with a predictable pattern. (Appendix, item 2.)

3. ***Differing transcription systems***. Languages written in non-Roman scripts may use multiple systems for converting names from native to Roman characters. One common Chinese name may be correctly written as either "Hsiao" or "Xiao" in its romanized form. "Chaiwat" and "Chaivat" represent the same Thai name. The same Russian surname may occur as "Ivanov" or "Ivanoff" or even as "Iwanow." The Soundex codes for the members of these spelling variants do not always match each other, so one form of the name will not reliably retrieve the others. Soundex was never intended to cope with the range of cultural diversity and orthographic complexity that typify enterprise databases in today's global economy, nor with the many standards used to convert names from their native written form into the Roman (A-Z) alphabet used by most computerized name search systems. (Appendix, item 3.)

4. ***Names containing particles***. Names in many cultures contain optional or supplemental elements that may be present in one instance of a name, but missing from the next. The Arabic name "Alhameed," for example, can also appear without the particle "al" as "Hameed" (or "Hamid," "Hamed," etc.). Both of these variants can refer to the same person. Soundex provides no means for accommodating such types of variation, but rather codes them as two different, non-matching names, with the result that two closely related variants do not retrieve each other. (Appendix, item 4.)

5. ***Perceptual differences***. When names in a database or text document collection stem from oral communication, many types of perceptual variation can influence the subsequent written form a name assumes. The Russian name "Tkachev" may, for

instance, be found as "Kachov" or "Tekacheff," since non-Russian speakers may not perceive the initial "T" sound consistently, or at all. Similarly, the name "Pfeiffer" could be mistaken for "Fifer," "Phifer," "Peiffer," "Pifer," or "Pipher." A query on "Pfeiffer" using Soundex, however, will not retrieve any of these potentially related names, nor will a variant like "Peiffer" retrieve "Pfeiffer." (Appendix, item 5.)

6. ***Silent consonants***. Soundex cannot capture the phonetic similarity between names with silent consonants and alternate or simplified spellings of those names in which these consonants are omitted. An uncommon name like "Coghburn" may be spelled as "Coburn," or "Deighton" may be recorded as "Dayton." Soundex assigns different codes to these pairs, guaranteeing that they can never match each other. (Appendix, item 6.)

7. ***Name syntax variation***. Differing name structures (models) are used by various cultures and societies. The familiar "first-middle-last" model used with many North American and Western European names fits poorly with names from many other cultures around the world. As a result, names may be mapped inconsistently into the fields of database records. The name "Mohamed Afzal Aziz" might be found in one database record with "Mohamed" as the first name, "Afzal" as the middle name, and "Aziz" as the last name. In another record "Mohamed Afzal" might be in the first-name field and "Aziz" in the last-name field, with no middle name. In still another record, "Mohamed" might be entered as the first name and "Afzal Aziz" as the last name. Soundex was not designed to deal with this type of variation in the form of a name. (Appendix, item 7.)

8. ***Name equivalence***. Some names have related forms that cannot be associated by any sort of compression or fuzzy-match logic. The birth records for a "Peggy Smith," for example, are likely to read "Margaret Smith." In parts of Asia, names based on Chinese characters may have completely different pronunciations across different dialects. For example, the names "Ng" and "Wu" are both written with the same Chinese character, and both may be used to refer to the same person under certain circumstances. Soundex provides nothing to aid in linking names which, though understood as being equivalent, are nonetheless written in very different ways. (Appendix, item 8.)

9. ***Initials***. Initials are often substituted for full names. Records for a "Michael Kissinger" and "M. Kissinger," for example, may well belong to the same person. However, a standard Soundex query on "Michael Kissinger" will not retrieve "M. Kissinger," and vice versa. (Appendix, item 9.)

10. ***Unranked, unordered returns***. Because its goal is to group names by assigning a common code, Soundex does not have the capability to measure the degree of similarity between a pair of names found within a group. This means that returns from a Soundex-based query cannot be ranked and presented best-first—names are typically returned as they are found in the database. Name variants that could plausibly refer to the same individual may be found well after names which are

clearly irrelevant, forcing the user to scan all returns from top to bottom. On a test query on "Deighton," for instance, the exactly matching surname was last in a return list of seventeen names, preceded by such unlikely candidates as "Desiyatnikov" and "Degaetano." A query on the name "Criton" returned eighty-seven names; the exact-match was eighty-first in the return list. (Appendix, items 10 and 11.)

11. ***Poor precision***. The Soundex algorithm reduces distinctions between strings of letters to such a degree that many obviously dissimilar names are typically returned for each search transaction. For example, "Courtmanche," "Corradino," "Cartmill," and "Cortinez" were returned on a query for "Criton." This superfluous information has very real costs for application designers, in terms of processing resources consumed, response times and user satisfaction. Worse yet, the precision degrades as the database size increases, for many typical Soundex applications. (Appendix, items 10 and 11.)

V. Improvements over Soundex

Several of the problems mentioned above are further exacerbated by the multi-cultural make-up of modern databases. John Hermansen (1985) notes that a fundamental problem for Soundex and its derivatives is that they are applied as a universal name-search method. An algorithm designed largely for English names is less well suited to handle names with sound patterns and structures as diverse as Arabic, Chinese, Thai, Hispanic, and Russian, to name but a few. No single algorithm that relies on a single mapping of sounds to letters can be expected to perform well across multiple linguistic systems, especially not when some degree of transliteration has been involved.

One important improvement in name searching was implemented in 1963 within NYSIIS, the New York State Identification and Intelligence System. A major innovation of this system is its culture-specific search methodology, intended to accommodate the large number of Hispanic names within the NYSIIS database. Based on their observations of the syntax and sounds of names within their database, the developers of NYSIIS created search techniques that allowed names with multiple formats and spellings to match.

For example, a query on the name "Rodrigues Y Vega Y Romano, Juan" produces the variant forms "rodriguesyvegayromano, juan," "rodrigeusvegaromano, juan," "rodrigues, juan," "vega, juan," and "romano, juan" (Hermansen, 1985). These forms are then processed by a modified Soundex algorithm and a sophisticated set of probability tables that are rarely implemented in systems that seek to imitate NYSIIS. This attention to the nature of the names in the databases to be searched enabled NYSIIS to attain precision and recall levels that exceeded what Soundex alone had been able to achieve. Nevertheless, the limitations of the NYSIIS algorithm and the increasing diversity of the cultures represented by the names in their growing data base forced New York to abandon the NYSIIS system in 1998.

The early operational success of NYSIIS showed that name search results can be improved if the search technology includes knowledge of the cultural particularities of

the names in the particular database to which it is to be applied. This success also raises the question of whether the Soundex methodology can be adapted for a particular database to improve results. Lait and Randell (1996) set out to answer just this question after finding that Soundex recall rates were disappointing (see above). Using the same database against which Soundex was tested, they progressively altered the Soundex code until the maximal rate of accurate returns was found, with the minimal increase in incorrect matches. The resulting algorithm was titled "Phonex." Phonex was able to return 51.79% of the correct matches in the database, as opposed to Soundex's 36.37%. While this is an improvement, it still leaves almost half of the correct matches undiscovered. Lait and Randell also note that neither corrupted data nor multi-ethnic data is addressed by their improved algorithm.

VI. An architecture for next-generation name searching

Reconsidering the eleven areas discussed above where Soundex has demonstrable limitations, we can now formulate a set of characteristics that an advanced name search system would need to possess, in order to meet the challenges posed by large, multi-cultural databases in which both predictable and random name-spelling variations are present in a significant number of records.

1. ***Culture-specific matching criteria***. Naming systems differ significantly from one culture to the next—in the relative order in which parts of a name appear, in the consistency with which they are written in romanized form, in the way they are abbreviated, in which parts are considered mandatory for identification. To accurately identify all potential matches, an automated name-search system must account for a name's culture of origin. Such knowledge will allow the correct set of matching techniques to be applied to the name. Ideally, such a cultural identification could be accomplished automatically, to add speed and consistency that humans cannot be expected to provide.

2. ***Automatic application of linguistic rules for the culture/language context***. This step may comprise a number of processes. A full name must be parsed, and possible word order variations and shortened forms may be generated. Spelling variants for each part of the name must be calculated. There are many possible approaches to this step—rule-based, algorithmic, statistical/probabilistic, or combinations of these. Furthermore, variants may be based on either phonetic (pronunciation) or alphabetic similarity

3. ***Noise tolerance***. Once culture-specific knowledge has been used to isolate and align those portions of the name to be compared, the character-level comparisons must take into consideration the possibility of random keying, which correspond to no orthographic or phonological principle.

4. ***Recognition of equivalent but dissimilar name variants***. In most cultures, names are found which are understood and accepted as interchangeable equivalents, perhaps

used in different social circumstances. Nicknames and pet names are prominent examples of given-name (first-name) variants in wide use among English-speaking and Western European societies. An advanced name-searching system cannot rely on matching only the "official" forms of a name, especially when many applications are tasked with merging data drawn from a wide variety of sources and formats.

5. ***Ranked returns, with the best matches presented first***. Matching names that are most similar to the query name should be returned before those that are less similar. A name search system must include a means to measure the degree of similarity between two names and rank them accordingly.

6. ***Statistical and probabilistic search aids***. Many advances in the field of Information Retrieval have special applicability to the problem of name searching. In particular, knowing the relative frequency of a specific surname within a particular population would allow a correspondingly greater emphasis to be placed on the discriminatory value of the given-name information in a search transaction. To use such statistical and probabilistic information effectively, it would need to be closely integrated with the matching and ranking logic of the search algorithm. This type of information becomes crucial when dealing with Korean names, since approximately 75% of the population share the top half-dozen surnames.

7. ***Syntactic flexibility***. Because names are particularly susceptible to misinterpretation when they are captured in electronic form from oral or written origins, differences in white-space placements or even field placements (within a database record) should be overcome to a reasonable degree in an advanced name searching system. In particular, Oriental names whose order is accidentally reversed and Middle Eastern names with prefixes mistakenly classified as middle names should be reliably and efficiently matched with their more standard counterpart versions.

8. ***Capacity for adjustment and tuning***. Name searching is a non-deterministic problem, meaning that it is not always possible to obtain definitive results. Practically speaking, one person's ideal search results may be regarded as poor by another person. Exact-matches are easy to identify, but there are many shades of similarity and equivalence possible to discern among related names, so "good" search results may depend more than anything on the linguistic and cultural knowledge of the user. Moreover, many collections of names are highly volatile, with a significant number of records being added and deleted on a continuing basis. This means that an advanced name searching system should provide numerous mechanisms for adjusting the quality and quantity of the matches it produces, so that a balance-point can be reached among the conflicting demands of speed, accuracy, and efficiency within any organization. There is no single best way to find all related names and no unrelated names in any name collection of reasonable proportions. An advanced system might also offer various self-test and calibration mechanisms, so that users and maintainers can converge reasonably quickly on a group of settings for all adjustable parameters that supports the mission, operational setting and business rules addressed by name searching transactions. Such utilities might also advise maintenance personnel

whenever the name collection has shifted enough in its cultural/ethnic characteristics to necessitate a recalibration, as when bulk updates are performed.

It is clear that Soundex -- and indeed its many name-grouping successors -- were not intended to address the range of name-related issues presented in the foregoing paragraphs. It is not so much that Soundex fails to deal with these problems; rather, Soundex does not contemplate such issues at all. As a result, they are either ignored in Soundex-based search systems (with undefined and latent risk for their owners), or else they are addressed in piecemeal fashion with custom application code that "wraps around" Soundex. Not wanting to become entangled in the many subtle complexities of name searching, many application developers simply follow a strategy of avoidance, reducing support for name-based searches or removing it altogether.

VII. Conclusion

This paper has shown that using Soundex as the basis for a name-searching application is both easy and potentially risky, especially for application developers. Perhaps the most costly result of relying on Soundex for searching and matching of name data is the potential for relevant records to be overlooked; more insidious, but equally problematic is the gradually worsening degree of precision exhibited in search transactions as the database size and complexity grows. Several studies have shown that a significant percentage of correct name-matches in test databases cannot be returned by Soundex.

For enterprises in which accurate data retrieval using names is a crucial aspect of one or more business processes, it will eventually become necessary to overcome the well-documented limitations of Soundex-based searches. The Phonex experiment showed, however, that improvements in retrieval rates are typically marginal, and may still result in search accuracy below required levels. As more and more custom application code is wrapped around the core Soundex search mechanism to mitigate its deficiencies, development and maintenance costs can grow quickly. More and more resources are required to provide consistent levels of user satisfaction and productivity, as database size and cultural complexity increase.

A more effective name search strategy must be designed from the outset for large, multi-cultural databases, must incorporate much more than letter-to-sound information, and must accommodate both random and predictable variation in the spelling of names, if it is to deliver consistent, accurate results as the data to be searched grow in size and cultural diversity over time.

Soundex has a seductively low entry-cost as a name-searching solution. It is free, it is easy to understand, and it is simple to implement, especially when the database to be searched is fairly small. Application designers should consider carefully, however, whether or not deferred costs for subsequent life-cycle maintenance and enhancements, together with the generally hidden risks of failed searches and superfluous matches for end-users might outweigh these initial benefits.

References

Hermansen, John C. 1985. *Automatic Name Searching in Large Data Bases of International Names*. Washington, DC: Georgetown University (dissertation).

Lait, A.J. and B. Randell. 1996. "An Assessment of Name Matching Algorithms." University of Newcastle Upon Tyne. Available at www.cs.ncl.ac.uk/research/trs/abstracts/550.html

Stanier, Alan. 1990. "How Accurate Is Soundex Matching?" *Computers in Genealogy* 3:7.

APPENDIX

The name searches demonstrated here were conducted on two databases using ANSI-standard SQL queries involving the Soundex function, as implemented in several leading commercial RDBMS products. The first database is a set of surnames compiled by the U.S. Census Bureau from 1990 census respondents. This database (referred to within this paper as Cen90) contains over 88,000 unique entries. The second database contains names from residential telephone listings for the 703 area code (Northern Virginia) from the year 1996. This set of names (referred to as Nms703) contains about 520K entries. A significant difference in the two databases is that Cen90 contains only surnames, while Nms703 contains full names (and sometimes household names) with the surname (family name) and the given-name(s) in separate fields.

1. SQL/Soundex, Cen90, Query = KORBIN

Initial-letter non-matches: The name "Corbin" is in the database but is not retrieved.

Matching Name	Name-ID
KARPINSKI	10341
KIRVEN	19606
KERVIN	20789
KERFIEN	23709
KRUPINSKI	28777
KARBAN	35497
KARPIN	37698
KARPINSKY	40342
KRIVANEK	43203
KURPINSKI	56099
KURBAN	56105
KIRVIN	56295

KORVIN	62734
KRUPANSKY	62672
KARPINEN	62949
KARPEN	62950
KARVONEN	71510
KRABBENHOFT	82633
KRUPINSKY	82545
KRIVANEC	82571
KARABIN	83136
KRUPPENBACHER	82544
KORBIN	82709

SQL query on CORBIN:

```
SQL> SELECT *
 2  FROM CENSUS90SN
 3     WHERE SURNAME='CORBIN';
```

SQL response:

1231 CORBIN

2. SQL/Soundex, Cen90, Query = SMITH.

Noise intolerance: The listing "SMITHJ" is in this database but is not returned.

Matching Name	Name-ID
SMITH	1
SNEED	1514
SMOOT	3138
SNEAD	3343
SHUMATE	3792
SMYTH	4106
SANDY	5477
SAND	6731
SANTO	7300
SANTOYO	7512
SMYTHE	7719
SMTIH	8345
SWINT	9276
SUNDAY	10290
SNODDY	10422
SINNOTT	10978
SMITHEY	11153
SMIT	12725
SMIDT	13405
SANTEE	13657
SHAND	13923
SANT	14771
SUND	15649
SMEAD	15660
SANDE	16742
SUNDE	18909
SUMMITT	18910
SANTA	18951
SMIDDY	20523
SANTI	21264
SANDHU	21266
SAINT	21927
SAMET	23424
SENNETT	23396
SANDT	26093
SAMMET	26095
SANDA	28387
SHIMADA	29653
SUNDT	31036
SAMUDIO	31205
SANDAU	32925
SANTOY	32919
SNIDE	32813
SANDO	32922
SMITHEE	34689
SINNETT	36826
SINEATH	36827
SMID	39190
SONDAY	42026
SANTOYA	42233
SINOTTE	42071
SYNNOTT	41910
SHINDO	42105
SANDOW	42238
SENNOTT	42137
SHANDY	42125
SANDOE	42239
SWANDA	45221
SANTAI	42235
SUMIDA	45235
SHINODA	45433
SENATE	45487
SHOEMATE	45431
SENATO	49465
SOMODI	49304
SHMIDT	49416
SEMIDEY	49467
SMIDA	49327
SONODA	49301
SAMIT	49623
SWANT	53938
SAMAD	54495
SENTI	60360
SANDI	60590
SNEATH	60161
SMIHT	60166
SNOWDY	60153
SAMIDE	60597
SANTY	60567
SHONT	60281
SENTA	68008
SANNUTTI	68285
SINDT	67824
SANDAY	68297
SMAYDA	67753
SAINTE	68348
SNITH	67734
SAINATO	68350
SMITTY	78004
SMITHE	78005
SOMDAH	77935
SANTIO	78754
SNODE	77989
SANTTI	78748
SAMMUT	78797

SQL query on SMITHJ:

```
SQL> select *
2  from census90sn
3     where surname='SMITHJ';
```

SQL response:

39189 SMITHJ

3. SQL/Soundex, Cen90, Query = XIAO

Differing transcription systems: The alternative transcription "HSIAO" is in the database but is not returned.

Matching Name	Name-ID
XU	10540
XIE	17912
XIAO	26915
XIA	34382
XUE	48597

SQL query on HSIAO:

```
SQL> select *
2  from census90sn
3     where surname='HSIAO';
```

SQL response:

35552 *HSIAO*

4. SQL/Soundex Cen90, Query = ALHAMEED

Particles: The variants "HAMID" and "HAMED" are in the database but are not returned. *Note: The variant "Hameed" is not in the census90sn database table*.

Matching Name	Name-ID
ALMEIDA	3153
ALMODOVAR	10536
ALLENDER	13870
ALAMEDA	14429
ALMEDA	21094
ALLINDER	23261
ALMADA	23260
ALLENDE	24951
ALLENDORF	32487
ALMETER	38705
ALNUTT	41491

ALMEYDA	38704
ALMODOVA	44731
AALAND	75675
ALUMMOOTTIL	75524
ALLNUTT	75543
ALLHANDS	75545
ALHAMEED	88631
ALAND	88660
ALAMEIDA	88662
ALEYANDREZ	88636
AALUND	88798

SQL query on HAMID:

```
SQL> select *
2  from census90sn
3     where surname='HAMID';
```

SQL response:

19656 HAMID

5. SQL/Soundex, Cen90, Query = PFEIFFER

Perceptual differences: The variants "PEIFER," "PEIFFER," "PIFER," "PEFFER," "PIEFFER," "PHIFER," "PYFER," "FIFER," "PIPHER," and "PIEFER" are all in the database but are not returned.

SQL/Soundex hits	
Matching Name	**Name-ID**
PFEIFFER	2766
PFEIFER	4240
PFEFFER	7308
PFEUFFER	45910
PFEFFERLE	45911
PFIEFFER	61210
PFIFER	79873
PFEFFERKORN	79875

SQL query on PEIFER

```
SQL> select *
       from census90sn
       where surname='PEIFER'
```

SQL response:

27279 PEIFER

SQL query on PHIFER:

```
SQL> select *
       from census90sn
       where surname='PHIFER'
```

SQL response:

4746 PHIFER

6. SQL/Soundex, Cen90, Query = COGHBURN

Silent consonants: The name "COBURN" is in the database but is not returned.

Matching Name	**Name-ID**
CASPER	2842
COSPER	7422
COGBURN	11257
CASPERSON	15240
COCKBURN	15886
CASEBEER	22390
CASPERS	23164
CHESBRO	25733
CHESEBRO	38390
CASEBIER	38413
CZAPOR	41022
CASPARI	44299
CASPARIS	47993
CASPERSEN	52520
CASPARIAN	58085
CHEESEBROUGH	58019
CHESBROUGH	58014
CASBEER	65110
CASBARRO	65111
CASBURN	65109
CHEESEBORO	74204
CASPAR	86956
CASPARY	86955
CZUPRYNA	86332

SQL query on COBURN:

```
SQL> select *
 2  from census90sn
 3     where surname='COBURN';
```

SQL response:

2356 COBURN

7. SQL/Soundex, Nms703, Query = AFZAL AZIZ, MOHAMED

Name syntax variation: A listing for "AZIZ, MOHAMED AFZAL" appears in the database but is not returned.

Matching Name	Name-ID
AFZAL, MOHAMMAD	3401
AFZAL, MOHAMMAD	3402
AFZAL, MOHAMMAD	3403
AFZAL, MOHAMMAD A	3404
AFZAL, MUHAMMAD	3405

Cf. SQL query on AZIZ, MOHAMED AFZAL:

SQL hits	
Matching Name	**Name-ID**
AZIZ, MOHAMED AFZAL	20529

8. SQL/Soundex, Nms703, Query = DEIGHTON, BILL

Equivalent names: Two listings for "DEIGHTON, WILLIAM" are in the database but are not returned. (The database also contains "DAYTON, BILL," but this is not returned.)

No SQL/Soundex hits

Cf. SQL queries on WILLIAM DEIGHTON and BILL DAYTON:

SQL hits	
Matching Name	**Name-ID**
DEIGHTON, WILLIAM	900056
DEIGHTON, WILLIAM	900055

SQL hits	
Matching Name	**Name-ID**
DAYTON, BILL	900057

9. SQL/Soundex, Nms703, Query = KISSINGER, MICHAEL
Initials: The database contains a listing for "KISSINGER, M.," but this is not returned.

Matching Name	Name-ID
KICHINKO, MICHAEL N	235497
KISSINGER, MICHAEL & DENISE	240408
KUZMIK, MICHAEL D DDS	249637
KUSHNICK, MICHAEL G	249445
KUZMIK, MICHAEL	249635
KUZMIK, MICHAEL D DDS	249636
KUZMUK, MICHAEL & ELIZABETH	249641

Cf. SQL query on M. KISSINGER:

SQL hits	
Matching Name	**Name-ID**
KISSINGER, M	240407

10. SQL/Soundex, Cen90, Query = DEIGHTON
Unranked, unordered returns: The query name is last in this return.

Matching Name	Name-ID
DUSTIN	7542
DISTIN	24738
DEGAETANO	30540
DAUGHTON	30548
DOUGHTON	32108
DIGAETANO	33923
DIGHTON	40920
DUSTMAN	43958
DESTINE	52180
DESATNIK	64616
DUSTON	73325
DESTINA	73612
DESIYATNIKOV	85996
DISTANCE	85868
DICKSTEIN	85932
DESTIME	85982
DEIGHTON	86161

11. SQL/Soundex, Cen90, Query = CRITON

Poor precision. Exactly matching name appears nearly at the end of the returns.

Matching Name	Name-ID
CARDENAS	781
COURTNEY	1273
CARDONA	2031
CARDEN	3533
CRAYTON	3723
CRITTENDEN	3976
CURTIN	4029
CURETON	5676
CARDINAL	7622
CARDIN	7693
CARDINALE	9050
CRITTENDON	10933
CARDONE	11100
COURTEMANCHE	11251
CORTINAS	11398
CHRETIEN	12082
CORDON	12246
CORTINA	18281
CERTAIN	20995
CROWDEN	22362
CARDON	22395
CARDAMONE	24840
CORRADINO	25721
CARTMELL	27892
CREEDEN	29155
CORDNER	29161
CARADINE	29226
CARTMILL	27891
CURTNER	29140
CARTHEN	29219
CARTON	30640
CRITTON	34000
CARTHON	32273
CARDENA	34099
CRATON	36040
CORTNER	36049
CARRADINE	38419
CARDIMINO	41162
CARTEN	41152
CORRADINI	41063
CARDINE	41161
CARODINE	41153
CARDANI	41163
CORTON	41059
CREEDON	44193
CHRITTON	44257
CROUTHAMEL	44184
CORDONE	47874
CARRETINO	52535
CARDENAL	52548
CARTIN	52530
CAROTENUTO	52536
COURTNAGE	52344
CRETEN	57839
COURTENAY	57857
CORDANO	57890
CARATTINI	58116
CARDNO	58108
CARDONIA	58107
CRATIN	64839
CARDINALI	65134
CORDONNIER	64886
CARDONO	65132
CRUDEN	64808
CORDONA	64887
COURTON	73972
CORTINEZ	73991
CORDENAS	74011
CORRIDAN	73996
CARTNER	74330
CARDINALLI	74351
CARDONI	87019
CARDENOS	87023
CRATION	86436
CORRIDONI	86494
CARRIDINE	86990
CERDAN	86891
CARIDINE	87017
CRITON	***86415***
CARTAN	86983
CHIRDON	86767
CARTHENS	86981
CARDINO	87021
CARDINAS	87022
CARADONNA	87030

