Introduction to Name Search and Matching

5th Edition

www.searchsoftware.com

The Math, Myth and Magic of Name Search and Matching
by Geoff Holloway and Mike Dunkerley

Printing History

November 1998	First Edition
September 1999	2nd Edition
May 2001	3rd Edition
March 2003	4th Edition
March 2004	5th Edition

Table of Contents

Chapter

1

Introduction

By Geoff Holloway President and Director of Research and Development for SearchSoftwareAmerica

When I decided in 1984 that “name search and matching” was a problem space worthy of a specialist company, I knew we would never completely solve the problem. Today I am still surprised at how complex and large a subject it is. Unlike many information processing issues, designers can not make the problem go away by redefining processes and procedures. The names and labels we necessarily use, in the data in computer systems, are those that are in common use in the real world, and we are not free to change them.

With our experience and tools, my colleagues and I have made the problem much easier to cope with, addressed many of the difficulties, and learned a lot.

We hope that this book will pass on many of the important things we have discovered and learnt.

About Search Software America

Whether it is names, addresses or other attributes, identity data suffers from unavoidable error and variation. Search Software America develops and markets software products that significantly enhance the ability of computer systems to find, match, group and investigate identity data. The SSA software technology is used by in-house developers, package vendors and systems integrators to deliver systems on Mainframes, Servers and Workstations. SSA's emphasis is on delivering the highest quality and performance together with ease of implementation. SSA's software is used in Customer Systems, systems supporting Identity Screening, Fraud Investigation and Compliance, in Data Warehousing and Data Integration applications, as well as in many other Commercial, Law Enforcement and Government systems. SSA has over 500 customers worldwide searching data from over 40 countries. Any country and character set can be supported. SSA operates as one worldwide sales and support organization providing 24-hour technical support, managed out of the USA, UK and Australia. For more information about our technology and services please contact us at one of our offices (located on the back cover) or visit us at www.searchsoftware.com.

About this book

The Math, Myth and Magic theme has taken a little dramatic license.

The Math of the data, technical solutions, and the results we measure, has reinforced our conviction that this is a persistent empirical problem. The Math tells us that this problem will not simply disappear with new technology.

Our exposure to Myth is the fact that we see continuing use of ideas and tools that do not work at all well. System users and builders have a mythical belief in many solutions because they are so frequently unaware of the data that they can not find. We attempt to fairly dispel these Myths.

And the Magic is the fact that strong solutions are becoming more and more cost effective and available, despite their necessary complexity. The significant reduction in the costs of storage, cost of compute time, and costs of redundant data allow magical solutions today that were not practical just a few years ago.

This book is of value to both the person who is tackling a "name search" or "name matching" problem for the first time, as well as to the experienced specialist in "identification and search" systems.

We have attempted to format the book so that it can be browsed, and that each section stands alone.

The rest of this chapter will introduce simple name search, matching and identification needs as well as some terminology. The remaining chapters contain short subject sections that have been loosely grouped under the chapter headings. Because the book is designed to be browsed some material will be discussed in more than one place.

What Naming Data do we use to Search with?

In many systems, computerized or manual, we need to find things that have been filed away using a Person's name, a Customer's name, a Company name, an Email address, a Place name, an Address, a File Title, an Author's name, a Book title, etc.. All such names are collections of words, numbers and codes that have been used to 'label' or 'christen' the original real world item.

In the real world we use these names in speech and writing as the labels for 'proper nouns' in sentences:-

Geoff Holloway lives at 17 Congham Drive, Pymble NSW

Holloway, Geoffrey Norman is the name on loan # 1256347

The Data Clustering Engine V2.01 is used by XYZ Co.

In systems and databases we use such names to find files, transactions, accounts, and any variety of data recorded about the 'entity' identified by the name or naming data.

Names are not normally unique.

Names when said, written and especially when entered into computer systems are subject to considerable variation and error.

You can not avoid this variation and error.

Even if the data on file is 'perfect', the 'search name' will come from the real world and be subject to natural error and variation.

What Identification Data do we use to Match with?

In addition to the words and codes in Names, Addresses, Titles and Descriptions, we frequently use other data to make decisions about whether we believe two reports or records are about the same entity.

SEARCH	ROBERT J. JOHNSTON	1962/02/12
	12 RIVER SIDE SPRINGVALE	(807) 2334 657
✓	BOB JAMES JOHNSTON	1962
	12 RIVERSIDE DR. SPRING VALE	2334 657
?	MR. R. J. JOHNSTONE	1962/12/02
	35 CITYVIEW CT. SPRINGVALE	1 807 4456 721
✗	ROBERT JOHNSON	1973/10/04
	2 MAPLE RD. BROOKFORD	555 763 2413

Data such as dates of birth, e-mail addresses, ages, phone numbers, gender and identity numbers are all subject to error and variation.

When a name is used to bring up candidates on a screen, people use all of the identification data returned to choose whether the records displayed are relevant or appropriate. In automated matching systems, the system itself has to be able to use the same data that people would use.

When people make choices about whether things match or not, they compensate for the error and variation. Our systems have to achieve the very same compensation that people make.

To confirm that records are in fact matching requires that our systems use the same data in the same manner as the human users of our systems would use. In fact our systems need to mimic our very best users doing the same job.

What Objectives do we have to satisfy in Name Search and Matching systems?

Whether the process is an on-line inquiry like Customer Identification, or a Criminal Records search, or a batch matching process like merging Marketing Lists before a selection for mailing, we must find all the candidates that could possibly be the same as each other, or are the same as our “search data”.

We must mimic a human expert in finding all the candidate records, and then make the same matching choices as the human expert would make for that specific business purpose.

This means that our searching and matching technology must overcome the natural error and variation that unavoidably occurs in all real world identification data. We must do this despite the fact that the process of capturing the real world data into computer systems actually introduces even more error and variation.

In many systems the objective is also to overcome fraudulent modification of identity data. This ‘class of error’ is more aggressive in that it does not occur naturally, but is introduced to defeat or control aspects of matching systems while retaining the defense that it was in error rather than fraudulent.

We will see throughout this book that any attempt to overcome error and variation increases the work done and therefore the cost. We will also see that, in order to compensate for more error, we always run the risk of introducing false matches.

The task is a balancing or tuning exercise between:-

- “Performance” and “Quality”
- “Under-matching” verses “Over-matching”,
- “Missing the Right data” verses “Finding Wrong data”.

Chapter

2

The Error and Variation

The variations that occur in names include spelling, typing and phonetic error; synonyms & nicknames; Anglicization and foreign versions of names; initials, truncation and abbreviation; prefix and suffix variations; compound names; account names; missing words, extra words and word sequence variations, as well as format, character and convention variations.

Variation examples

Person Names

- William, Bill, Billy, Will
- Chris, Kris, Christie, Krissy, Christy, Christine, Tina
- Franc, Frano, Frank, Francis
- Peter, Pete, Pietro, Piere
- Johnson, Johnsen, Johnsson, Johnston, Johnstone, Jonson
- Smith II, Smith jr, Smith 11, Smithjnr
- De La Grande, Delagrande, D L Grande
- Henry Tun Lye Aun; Mr Aun Tun Lye (Henry)
- Frank Lee Adam; A. Frank Lee; Lee Frank
- Patricia Jane Morris; P J Morriss; S. F. & P.J. Morris

E-mail

- michael_smith@searchsoftware.com; mike.smith@searchsoftware.com;
- jbrown @aol.com; john.brown@yahoo.com; jb2599@optonline.net

Companies

- B. Lamond Inc.; A & B La Monde Co; AB Lamond Incorporated; Lamond Inc.
- International Business Machines; IBM; I.B.M.; Intnl. Bus. Machines
- Stanley Rutherford & Associtesl; Messrs. Rutherford, Stanley and Assoc; Rutherford Assocs
- Abe Goldberg & Sons; Abe Boldberg and Son; Abe Godberg & sons;
- Aba Din Inc; Mitchell Holdings dba Abadin Inc
- Abbotts; Abbots Accounting Services; Abott's Accountancy; Abbots Accountancy Advisory Svcs
- Virginia Trust Company; Trust Company of Virginia

Addresses

- Jackson Rd. East Hartford; 117-2a Jackson Rd East, Hartfrd; 2a East Jackson, Hartford; 117a Jackson Rd, E. Hartford
- Ground Floor 192 Aberdeen St South Head; Grd. Fl. 192 Aberdeen St Southhead; 192/1 Aberdeen Sreet Sth. Hd.
- Suite 9A, The Russell Center, Washington Plaza, New Haven; Room 9-A Rusell Bldg, Newhaven

Dates

- 12/14/1998; 14/12/98; 14th December 1998; 14th Dec 1998; December 14th 98; 1998-14-12; 98/12/14.
- 7/2/1996; 7/2/9600; 7/2/96.

Phone Numbers

- 900-869-1481; 90-08-00-86-91-481; 8691481 ext 67; (0) 7778691481; (+44)777 8691 1481; 869 1481.

Null Values

- Not Known; Unknown; Missing; DOA; John Doe; Baby Doe; Jane Doe; Corpse; XXXXXX
- No Middle Name; No Initial; NMI; NMIK; Nomiddlename.
- 00-00-00; 99-99-99; - - -;

Missing words, extra words and word order variations

Each usage of a person's name will normally contain several pieces of its essential identifying components. However over time and across systems, the pieces present, and the forms used can differ greatly.

It is not unusual for a name like William John Brown to appear over time or in different databases as William Brown, John W Brown, Bill Brown, W Brown, W J Brown, William John Henry Brown or some other variation. Certain common 'extra' words like Senior, Junior, Doctor, BSc, Phd can appear quite frequently in a name. Infrequently, anything can appear in a name field e.g. e-mail addresses, unusual titles like My Friend or Maestro, or unrecognizable foreign language forms of common title words.

A common reason for extra unexpected words in names, addresses and other data is that the real world words were simply too many to fit the available space. This can lead to the user or the data entry system placing them in the beginning of the next field.

Missing and extra words are also common in organization names and addresses. For example, the Plaza Hotel may be known as the Town Plaza Hotel, the Plaza Hotel NorthSide or the Plaza ParkView Hotel.

Words appearing in different orders are also very common. The Plaza Hotel may be known as the Hotel Plaza Northside. Say Chi Tong may be found as Chi Tong Say, Chi Tong, William Say (Chi Tong). There is no guarantee that John Smith will not be recorded somewhere as Smith John, or in fact, Smith could be the person's first name.

When designing systems, databases and keys, a strong solution will always assume names are incomplete, out of order, that ‘noise words’ are present, and that it is impossible for programs or people to ‘clean’, ‘reorder’ or ‘remove all the noise’ in names to make them ‘correct’.

System Introduced Errors

Computer systems frequently introduce error into identity data by making wrong assumptions about the structure and meaning of the data.

An assumption that the family name will never be more than 15 characters, truncates long names (some foreign naming words are unusually long).

An assumption that all people enter their name in a stable order will capture an incorrect sequence (some cultures present family name first, rather than last).

An assumption that an address will contain street information before locality information does not work for all countries.

The following foreign name and address in a USA customer database

```
William Ho Kwok Ki ,
10/F Cityplaza 4, Taikoo Wan Road,
Taikoo Shing, Hongkong 123
```

could be badly parsed by a US formatting program into:

Family name:	**Ki**	First name:	**William**
Middle names:	**Ho Kwok**		
Building:	**10/F Cityplaza**		
Street:	**4 Taikoo Wan Road**		
Town:	**Taikoo Shing, Hongkong 123**		
State:	****unknown****	Zip:	****unknown****
Country:	****unknown****		

In fact, the family name is `Ho`, the street address is `12 Taikoo Wan Road` (the 12 was accidentally omitted and the 4 from Cityplaza 4 incorrectly used), and the town is `Taikoo Shing`. The country name

wasn't identified because the program only understood the form `Hong Kong`, and the postcode did not match the US format expected.
In reality none of this detail knowledge should be necessary to find or match this record.

It is easy to see how this record could later be missed when the search street address was `12 Taikoo Wan Road`, or the search name was `William Ho`, or `Ho Kwok Ki`, or `Ho Kwok Ki (William)`.

When data is being acquired from outside sources, as is frequently done for marketing systems, it is often necessary to compensate for the errors introduced by the original owners of the data. If data has previously been cleaned or scrubbed before it arrived in your system one can encounter errors such as `10 Jackson St Johns Wood` as having being enhanced to `10 Jackson Street, Johns Wood` when it should be `10 Jackson Court, Saint Johns Wood.` Or such names as `Bobby Jones` having been enhanced to `Mr. Robert Jones` when it was in fact your good customer `Ms. Bobbie Jones.`

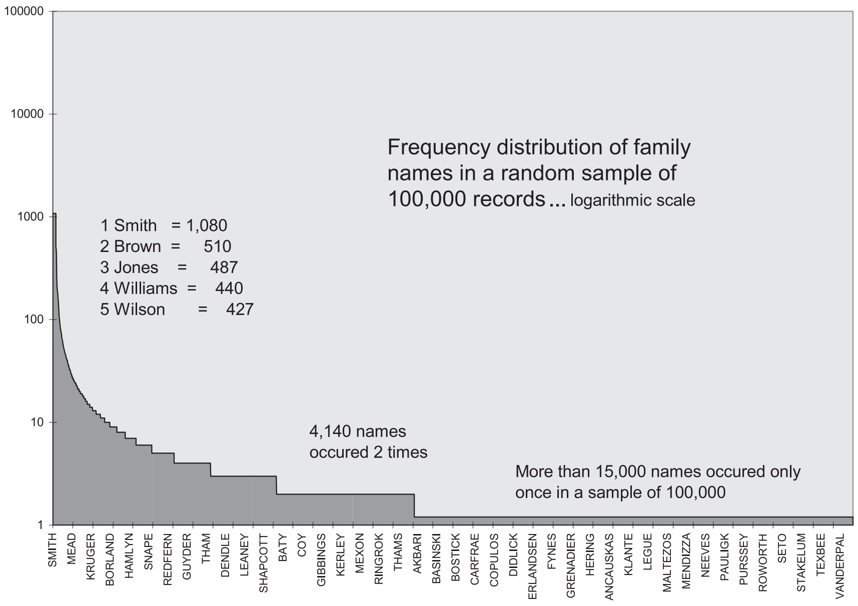
Whether parsing into fixed fields is done by people or by software, there will always be parsing errors or data that can not be parsed.

Search and Matching Systems must be designed to work well on data regardless of its shape or order, and without a need for detail parsing or understanding of the components.

Common and Uncommon Names, the Naming Word Vocabulary

The words we use to label things are chosen from a very different vocabulary than meaningful language. There are no dictionaries, spell checkers or rules for the names of people, places, things or even addresses. The vocabulary in use for people's first names includes in excess of 2,500,000 words in the USA alone, yet as much as 80% of the population may have names from as few as 500 words.

Family names are just as badly distributed.



Name search and matching systems must work well at both ends of this extreme curve. It must perform for the uncommon names as well as for very common words. This is an extremely difficult challenge when a database of 100,000,000 people may contain 100,000 John Smiths or 1 Main Streets.

Chapter

3

Popular (but not so hot) Techniques

There are many techniques in use around the world to address the need for name or address search and matching. Some are quite basic and unreliable, some work well for the average search, and some are quite sophisticated. An organization that is serious about the quality and performance of name search should understand what types of names a particular technique will miss and analyze the alternatives.

Exact Name Searches

The common use of exact name keys is quite absurd; it leads to much of the duplication of records, accounts and customers in systems.

It is not true that if you find an exact match you have found the right record. It is not even true that an exact match is a 'better match' than one with some variation. The real point of a search is to find the records that could practically be considered a match, or to find all the candidates that a user could believe were the same person.

Variation and error is so common that all searches to find the 'right' record, using names and address and other identification data, must be built on selection techniques that overcome the error and variation and find every one of the 'good' candidates.

Wild card searching on names

Simple wild-card searches are often of the form;

```
Search for  JOHN*       Search for  *NIGHT*

            JOHN          Returns:  NIGHT
            JOHNY                   KNIGHT
            JOHNSON                 ALLNIGHT
            JOHNSTONE               KNIGHTLY
```

Wild-card searches do overcome SOME error and variation in the name. That is why users believe in them. If the users knew what data they were missing they would not believe in them. Programmers believe in them because they learned to program them in their computer science classes at university.

In reality they only work if the searcher correctly guesses the right characters to include or exclude, and the database had no error in the characters actually used.

They can not find all the relevant records, do not address nicknames and abbreviations, or the fact that different records have different errors.

Wild-card searches normally return too many irrelevant candidates and will always miss many of the relevant candidates.

Wild-card searches are not for the serious searcher.

Keying partial words to save time

The idea that data entry time can be reduced by only keying partial words does save time e.g. the first 3 characters of the first word in a name followed by the first three characters of the last word.

It however makes it completely impossible to use techniques to overcome variation and error in words in the database. Such techniques as Soundex or techniques to handle nicknames or Anglicization, or translation or formal abbreviation, can not be applied to partial words. These techniques will only work well on complete words or their recognizable formal abbreviations.

Instead of keying partial words, key fewer, but whole words, or a combination of whole words and separate initials.

Text Retrieval Software and Name Search

The idea that text retrieval packages can successfully be used for name search applications is only believed by users who are unaware of the 'good' data that they miss.

Even when "Full Text Inversion" indexes have phonetic algorithms or 'expert' rule bases, for Name searches the indexing mechanism will result in an inefficient process. Imagine the cost of finding all the index values for the records containing John and then joining them with those that contain Smith to discover the subset John Smith.

For such packages and systems to be successful they must first recognize, find or discover the 'name phrases' in the text and index them with the same specialized techniques that are necessary for quality and performance in any other name search system.

Typical free text indexing techniques do not allow high performance or high quality retrieval of data using names.

Match-codes

A Match-code is an entity key built from a combination of the entity's attributes.

A simple match-code for,

```
John Smith
dob: 22 Feb 1979
2/234 33RD St., New York, NY12345
```

might be: **SMITHJ79NY**

Some match-coding aims to build a unique key for entities, and could add a sequence number to the end.

Match-codes require that the entity is first strictly formatted into its pieces; that all pieces used are in the 'stable' order; and that there are no errors in the pieces used.

If the above was always true then Match-Codes would be successful. Typically Match-Codes do find "correct records", but they frequently miss all the "other good candidates".

If codes are part of the key (e.g. dates, state/postal codes) the code must match exactly or the search will fail.

If there is no ability to overcome error and variation in the name parts, all records with such error and variation will be missed.

Missing word, extra word and word order variations are often missed unless the searcher permutes the search details.

Match-Codes fail if any one piece of the data used to build the code is not identical.

Match-Codes are not for the serious search or matching application.

Yesterday's Soundex's and related tools

In the early 1900's the Russel Soundex, was developed to provide a stable manual filing code for the USA Census documents. The development of this algorithm for encoding the last name of a person was based upon phonetics and certain classes of typical spelling and filing errors.

It was a simple set of rules to convert a last name word into a four, five or six digit number that had a high probability of being the same for two words that were variations of each other.

Since then, many Algorithms with a similar objective have been developed and modified. In 1969 the New York State Identification Intelligence System project evaluated the then popular algorithms.

This included evaluation of last names on New York State criminal records to see how effective algorithms were at overcoming the error and variation. To do this, they used records that were previously paired using fingerprints.

They evaluated such algorithms as: Soundex and many of its variants; LA County Sheriff; Consonant coding; Phonic standard and extended; Michigan Lien; and several Extract list based systems. The end result of their project was to define a popular algorithm known as NYSIIS which they proved was better optimized for their data at that point in time.

At Search Software America we have conducted many such exercises on data from many customers and countries and have found that, even in one country, no single algorithm is optimum for all naming data.

Fundamentally the choice of the optimum word stabilization algorithm depends upon the source of the data and the frequency distribution of certain classes of error in the data. For example carefully written and captured Criminal Records have a very different error distribution than urgent verbal requests for information over a radio. It is also true that written applications for a Bank Account will have different error and variation than phone orders for a magazine.

Whilst such stabilization algorithms can be a critical piece of a name search engine, the algorithms of the past are not suitable for use as database keys with today's data or volumes. Typically they either cause too many records to be found, or they miss too many relevant records.

Where stabilization algorithms are used, the choice of the right algorithm can be critical and must be based upon the true distribution of the error and variation in the both the population of file data, and the population of the search data.

The objective of word stabilization algorithms should be limited to retain as much of the original words form as is commensurate with the objective of "not missing valuable records".

The design of suitable database keys to maximize quality is a separate and much more complex exercise.

Some historical references follow:

- 1000 Division Code for Proper Names — IBM 1935
- The Computable Name Code — Niemoeller, Ralph K. 1958
- Phonetic Frequency Coding — Bluett, Fred S. IBM
- Identification Techniques — IBM 1969
- Name Search Techniques (New York State Identification and Intelligence System) — Taft, Robert L. 1970

Combining attributes like Sex, Birth Date, State or Postal Code with Name Search keys

Attributes such as Sex, Birth Date, State, Postal or Zip code have a stable, known set of valid formats and values and can be accurately edited and validated. However the fact that they are valid does not mean that they are true, accurate or consistent.

Mathematically such data can be ‘precise’ but not necessarily ‘accurate’.

If you use such data in keys, then the search is faster and fewer candidates will be found. However, if in either the file or the search data the attribute is <u>not true</u>, then <u>you are guaranteed not to find the record</u>.

If as sex code is captured as M or F, or 1 or 2, or as pictures of stick people that are then captured as codes, the data will be precise. But this data will have a rate of error (% that it is not true). Nothing will overcome this error if the sex code is used in a key or search argument to a database.

If the error rate in capturing the sex code is 5% in the file and also 5% in the search data, and it is included in the key used for a name search, then approximately 10% of the time your name search is forced to fail.

Error rate in such data is frequently high, as there is no redundancy in the data and no knowledge of its credibility.

A keying or writing error in such data can never be discovered or compensated for if it is used in keys.

With dates it is possible to discover an incorrect format, but like the sex attribute there is no way of discovering that a correctly formatted date is in error. So using dates in name search keys guarantees at least failure whenever the date is not true.

Another major problem with dates of birth is that people typically use multiple dates in identification data. This quite innocently arises because of the fact that many identification documents themselves contain errors, and after the first time a person tries to use a passport or driving license that has an error in their date of birth, and gets into to trouble with their identity, then in future they always quote the erroneous date of birth.

Less innocently it arises because of the prevalence for teenagers to acquire false identity documents early in life to overcome age restrictions on such things as entry to clubs, employment and drinking age.

Attributes such as State codes, Postal or ZIP codes and other simple geo-coding constructs is again a problem of truth. Much effort has been put in to systems to confirm that City/Suburb/Town names are consistent with State names and Postal codes. Whilst this may discover an inconsistency, the process of correcting it is frequently arbitrary or done by unreliable operators, and itself introduces things that are not true.

Modern data capture methods and uses of data enhancement technology are themselves responsible for a lot of consistent but untrue data.

Many systems rely upon the operator keying a postcode and having the city/suburb/state automatically generated. If a postcode is keyed incorrectly the result is consistent but untrue data. This also arises when operators choose from a list as they frequently select an entry one above or one below the intended entry.

Because lack of truth is common in simple attributes such as sex and state codes, dates or postal codes; and because the variations cannot reliably be compensated for; designing search keys which contain such attributes is very weak and will result in many missed matches.

Using such keys to partition a name search file should only be done if extreme volumes create performance demands, and the true quality of the attributes is measured, and the business penalties arising from missing useful data are clearly understood by all users.

However, attributes such as Sex, Birth Date, State, Postal or Zip code, are extremely useful as 'supportive' matching data used separately from the search keys.

They can be used in matching to help refine or rank a list of candidates found because, in well designed Matching, any element can be untrue or missing and the other elements can make up for it.

Cleaning, Scrubbing and Rejecting Invalid Data

There is a sound business reason for attempting to make sure that the data that is captured and stored in computer systems has the maximum value to the business.

There is also great value in having the data in easy to process shape.

However there is little value in having easy to process data that has become untrue as a result of the shape it has been transformed into.

The thrust associated with Cleaning, Scrubbing and Transforming data, once it is in a computer system, and the thrust that says lets only store ‘valid’ data suffers from a large number of pitfalls:

- Much of the data about transactions is legally necessary data and can not be changed without approval of the customer.
- Statistical techniques for enhancing data, are ‘good for statistics’ but introduce error that is destructive for matching.
- Many cleaning techniques are not reversible e.g. changing Bobby to Robert; changing St to Street when it is possible that it could be Saint.
- Users believe that the transformed data is true, and base decisions on it.
- Merging two records can lead to loss of data if later you find it was in error and they should be split.
- Rejecting invalid data, simply means it can not be used for anything and all business value of that data is lost.
- Cleaning projects conducted by people, suffer from the normal inconsistencies that arise with all other data.

For maximum truth in search, matching, and identification, work with and keep the original data in its real world format as originally entered.

In moving an obviously erroneous date field like 11/33/1967 into a data warehouse, or off an application form into a database, unless it is possible to go back to the real world source and get the true value, the best you can do is store the data exactly as it arrived. From a search, matching or identification point of view, a later transaction for a similarly named record with a date of 1/3/1967 or 11/3/1967 or 11/13/1967 or 11/23/1967 can be easily matched to the above date.

If such an invalid date is left out of the data warehouse, or converted to a 'null' or 'unknown' format the value of the data is entirely lost.

Aggressive validation of such data simply leads to users inventing 'valid' data. If you do not believe this, simply run a histogram on the date fields in a database—you are likely to find that the 1st of the month is abnormally common, and in really old data you may even find that 01/01/99 or even 99/99/99 is common.

As error and variation is quite normal in real world data, systems must be designed to retain the original data and continue to work despite this unavoidable error and variation.

Chapter

4

File and Field Design Issues

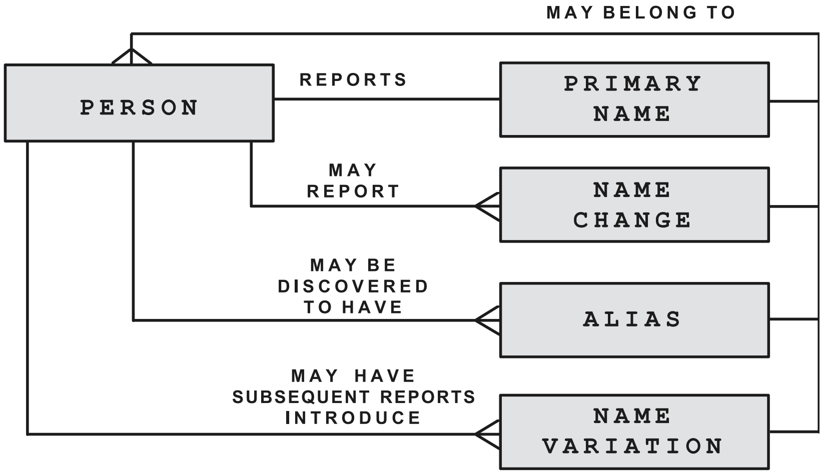
The design of file and field structures to support reliable name or address search and matching requires a good understanding of the nature of the data.

More Than One Name Field in a table or file, Names are truly "Many to Many"

It is obvious that two people or companies, or products, can have exactly the same name. It is also obvious that, even ignoring error and variation, people places and things have more than one name:

- People have maiden names and married names;
- People have aliases and professional names;
- Companies have registered names, trading names and division names;
- Places have several addresses, on two separate streets, old addresses, billing addresses, postal addresses etc.
- People and places can have names in more than one language.

The relationship between a name and that which it names is quite naturally a true "many to many relationship".



It is not surprising that indexing these “many to many” relations requires careful design in the majority of today’s relational databases, whose constructs are limited (with some good reason) to architectures based on “one to many” relations.

The design of a record or row that contains two fields, one for “name” and one for “maiden name”, or “registered name” and “trading name”, may make logical business sense, but it is not good for indexing.

When we are searching for a person’s name, company name or address we do not know which ‘role’ it plays. We do not know if it is a birth name, married name or maiden name, we do not know if it is a current or prior address. In order to address this problem effectively, it is necessary to have several index entries pointing to the same record.

Solving this “many to many” characteristic of names leads to an additional table or file in most databases. It therefore requires that this table is maintained in sync with the main business tables.

The Name Change Transaction

Whilst it is arguably necessary that whenever you have a name field in a system, then there will be a “name change transaction”, great care must be taken in deciding what to do about a name change.

In most cases the need to change a name will arise because a new transaction about the same person or company or product has been encountered. Another case is when the person has changed their name as a result of marriage, divorce, preference or simply discovered that you have it ‘wrong’.

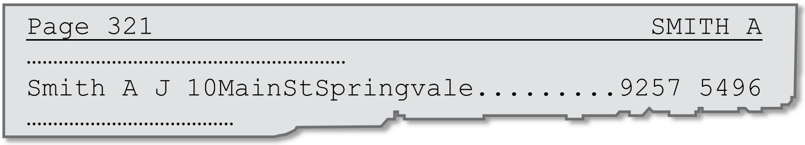
Usually removing the “old name” from the system is a bad idea; simply keep it as a known alias. References from “old documents” are very likely to create searches about “old names”.

Every name you encounter about a person, place or product is clearly evidence that rightly or wrongly that name is in use or has been in use in the real world about that same person.

To maximize your ability to find or match this entity in the future, the strongest way to deal with name changes is to add an additional name to the index for the same entity. For business reasons it may be necessary to identify one name field as the preferred, current or registered name.

The Telephone Book as Metaphor for Name Search Index Design

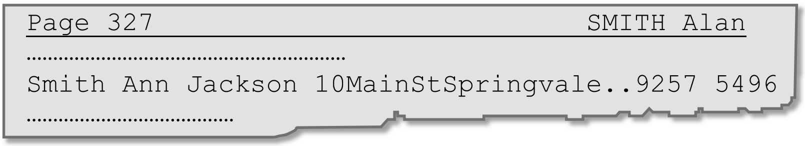
In the telephone book, a search for the name **Ann Jackson Smith** would normally succeed, on the "**Smith A**" page.



```
Page 321                                        SMITH A
.........................................................
Smith A J 10MainStSpringvale.........9257 5496
.....................................
```

When the name being searched for is **A J Smith** or **Ann Jackson Smith**, the entry is found relatively easily by browsing through all of the **Smith A J** entries.

A search for **A Smith** or **Ann Smith** is slower because more names must be browsed. If the full name had been indexed, the search for **Ann Smith** would be faster and the search for **Ann Jackson Smith** even quicker.



```
Page 327                                    SMITH Alan
.........................................................
Smith Ann Jackson 10MainStSpringvale..9257 5496
.....................................
```

Though this increases the size of each entry and the cost of capturing the information, the overall performance of searches is improved when there is more data in the name.

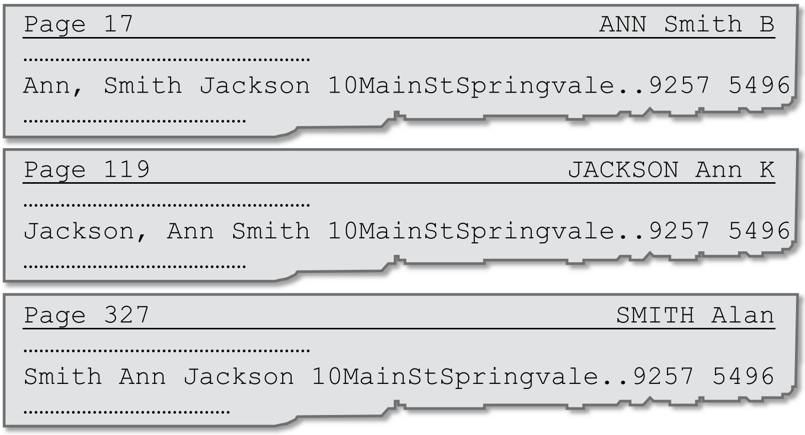
Given a full name to search with, its entry can be found more quickly.

In addition, when the name being searched for has missing or extra words or words in a different order, the simple telephone book indexing system starts to break down.

Searches for **Ann Jackson-Smith**, **Ann Smith Jackson** or **Smith Ann** will fail unless the searcher, after failing on the "**J**" and "**A**" pages, permutes the words and looks on the "**S**" page.

Regardless, a search for **`Ann Jackson`** will never succeed if the entry in the book was **`Smith, J.A.`** or **`Smith, Ann Jackson`**

If, however, the name **`Ann Jackson Smith`** was indexed on three pages of the telephone book, on an "**`Ann`**", "**`Jackson`**" and "**`Smith`**" page, by permuting the order of the words, then any of the above searches would succeed by opening one page.



```
Page 17                                 ANN Smith B
...............................................
Ann, Smith Jackson 10MainStSpringvale..9257 5496
.....................................
```

```
Page 119                              JACKSON Ann K
...............................................
Jackson, Ann Smith 10MainStSpringvale..9257 5496
.....................................
```

```
Page 327                                 SMITH Alan
...............................................
Smith Ann Jackson 10MainStSpringvale..9257 5496
..................................
```

The size of the telephone book increases, but search cost does not. The extra 'index entries' increases the physical size, yet improves overall quality and performance because any one page succeeds.

In computer databases, with today's low data storage costs, regardless of the volume of the file, the right solution for name indexes is permutation of words in the index entries at update time. This involves storing multiple records on separate 'pages' in the database just like our example in the telephone book above.

Permutation of naming words at search time alone can not guarantee to overcome the missing word, extra word or gross single word errors. This is not a design problem that can be overcome with better design, it is a mathematical constraint.

File Design for Optimal Name Search Performance

A search for all records that are relevant candidates for one set of search data, requires that one must display a list of good candidates on a screen or present this list to a batch matching/selection program.

To achieve this, the search data will be computed and used to Find, Read or Select a range of candidates from the database. This may be one or more logical requests to the database (e.g. several 'Select', or 'find' statements, may be necessary).

The database size affects the average number of candidates in a given range. The bigger the file - the more candidates are on file

dB Size	Ellen Dodd	John Smiths
100,000	4	50
1,000,000	40	500
10,000,000	400	5000

Searches are usually distributed the same way–if John Smith is .05% of the file; it is normally also .05% of the transactions.

The on-line name search transactions logically require:

- Computation to build search ranges based on the data use used in the keys;
- Physical access to the database to get index entries;
- Physical I/O to retrieve the display and matching data for all candidates;
- Computation to eliminate, ranks, or sort before display.

The time consuming work is the physical I/O:

- One or more physical I/O per Index entry per logical database command;
- One or more physical I/O per block of candidate data records;
- <u>If 'joins' are necessary </u>to get complete data for Matching & Display - <u>more than one</u> physical I/O will occur <u>per data record</u>!!!

The only way more than one candidate can be in a physical block is if the database file or table is ordered in the name key sequence. Even if this is true, little advantage is gained if access or “joins” to other tables are necessary to complete the display of a candidate line. To achieve acceptable response time, all display or matching data for a candidate must be in the same record and candidates must be in physically adjacent records. This can be achieved very effectively into today’s database systems with ‘index-only’ tables.

Achieving acceptable response time for even a single screen of candidates can not be done if each line requires multiple physical I/Os.

You can reduce the number of candidates or screens by automating the choice, selection or matching process, but the data still has to be read from the database and presented to a “matching” program, so the need for physical optimization is still very necessary.

Of course the average number of candidate records read should be kept to a minimum, but this minimum will relate to the size of file, how common the name is, and to what degree it is important not to miss possible matches. This decision should be tied to individual transaction and business risk/benefit.

To get good response time in name search, de-normalizing & maintaining a copy of the relevant name search & matching data in optimum physical sequence is essential.

Coping with a small % of Foreign Name & Address Data in your files

With today's electronic communications, WEB based marketing, and global business environment, it is inevitable that some prospects and customers in a local or regional file will have addresses from other countries. The percentage of this data in your files may be small but it is growing, especially in prospect files that are purchased or rented.

A common problem in coping with such data is thinking that rigid local standards can be made to work for this foreign data.

Asking the input data to be formatted into detailed fields according to strict local rules is inviting assumptions and choices that can vary from person to person, country to country. This leads to country name in state fields or postal code fields, apparently invalid postal codes, postal codes in address line fields etc.

Requesting input in unformatted or loosely formatted fields is the best way of obtaining reliability and completeness. If transaction and file formats for names and addresses are designed like the lines on an envelope you will be able to capture both local and foreign data with maximum integrity. This will mean that the search and matching system should be designed to cope with unformatted data. Systems can be reliable in dealing with unformatted data; people are not reliable when they are asked to format data.

This approach is essential for multinational systems but also very relevant for maximum value in local systems.

Don't try to overcome these problems before the data is stored. Let the system overcome them. Use simple large fields for names and addresses that allow users to input data as they would on an envelope or business card.

When partitioning keys makes sense

It is a misconception that partitioning search keys improves the reliability of a name search. Partitioning will always result in some loss of reliability.

However, all name search systems are susceptible to a conflict between performance and reliability. When extreme volumes of data are to be searched, and performance is more critical than reliability, there is a case for partitioning the keys.

The choice of what data to partition with also creates a conflict between quality and performance. An attribute that achieves the performance objective, but is not measurably reliable, is not helpful. An attribute which is measurably reliable, but which does not meet the performance objective, is also not helpful.

For some systems a year of birth may be a good partition, but no good if the error rate in birth dates is high. For other systems a state code may be a good partition, but no good if there is a high rate of movement between states, or a lack of truth in the state codes.

The need for partitioning should be empirically derived (as a result of tests on real data, in real volumes, in a production-like environment) and not decided upon theoretically.

If partitioning is used, when null or suspicious values of the partitioning attribute are encountered, these must be added to a common partition which is searched whenever a specific partition is searched. Also when nulls or errors are found in the search data's partitioning attribute then all partitions must be searched.

A strong search system will allow searches across all partitions, even if this is not the default search.

Storing the good with the bad

In many business and government systems it is necessary to index data about both the 'good guys' and the 'bad guys',

- Customers or Accounts, rather than ex-Customers or ex-Accounts who have Bad Debts or for whom Service is Denied;
- Prospects, rather than Do Not Mail names;
- People being protected, rather than the Terrorists and Trouble Makers;
- Persons with Petty Criminal Records, rather than Dangerous Criminals.

While the data stored may be identical, this is not a good reason for storing the information in the same file. If they are stored together and indexed together it is easy to miss a critical 'bad guy'.

In many system designs, a central name index or personality file is created with one common Name Search dialogue built for it. Then simply because it exists and contains names, addresses, account numbers, and other identity data together with system references, all forms of data are stored in this one 'cross reference' index.

For both system performance and quality, and to allow user dialogues to be more efficient and effective, the records about negative or risk related information should be indexed separately using more exhaustive and expensive techniques for the negative data. Certainly the commonality of the process and formats can be taken advantage of by sharing code and inheriting designs, but mixing the good with the bad is never a strong design.

In order to maximize the chance of finding the high risk 'bad guys' keep them in separate files, index them more exhaustively, and use more exhaustive search strategies.

Chapter

5

Performance and Quality Issues

The task of developing name search and matching systems is a balancing act between:-

- *“Performance” and “Quality”*
- *“Under-matching” verses “Over-matching”,*
- *“Missing the Right data” verses “Finding Wrong data”.*

Effect of file size on name search performance

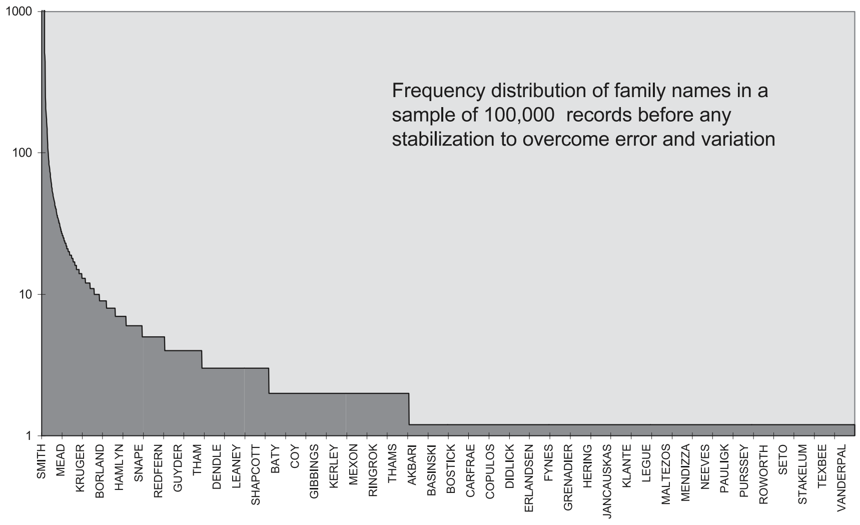
Because there is an extreme skew in the distribution of words used in people’s names, company names and addresses, some names will cover many candidate records, while other names will have only a few candidates.

If SMITH represented 1% of the population and Lebedinsky .001%,

Population Size	Number of SMITH’s	Number of Lebedinsky’s
1,000	10	1
100,000	1000	1
1,000,000	10,000	10

If the family name was used as the name key, a search for SMITH in a 100,000 record file would be slow, in million record file, prohibitive.

The following diagram shows the typical skew of names in a population.



Evaluating Stabilization Algorithms

In order to overcome phonetic and orthographic error in names, it is necessary to apply some form of transformation to the characters in the name. Stabilization Algorithms transform characters which are commonly confused with each other, into a common form.

For example,

PH	=>	F
M	=>	N
A,E,I,O,U	=>	A

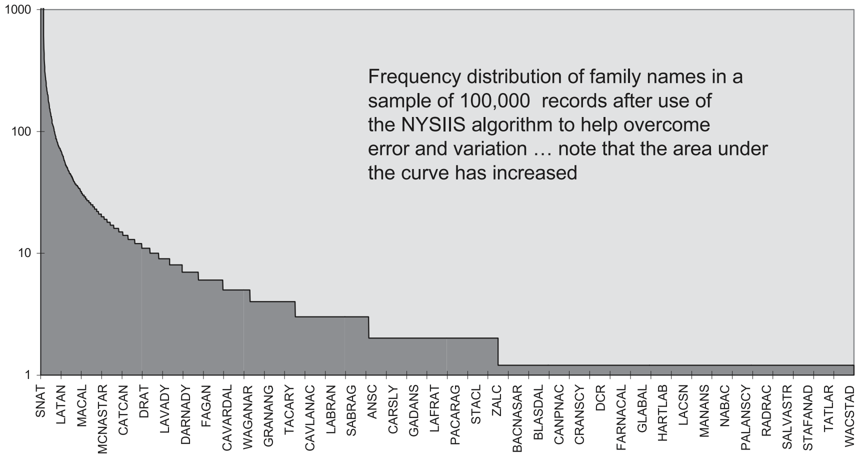
There are various formal Stabilization Algorithms available, many of which have been further customized by individual user organizations.

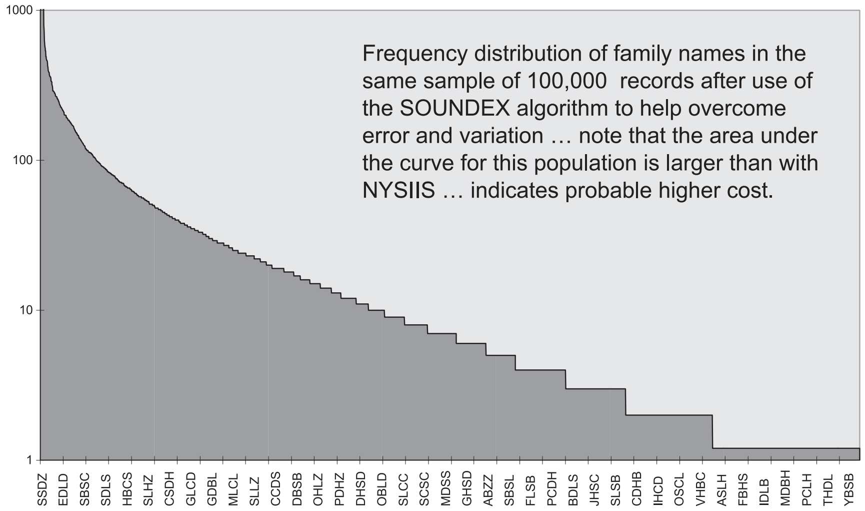
The one thing that all such Stabilization Algorithms have in common is that their application aggravates the frequency distribution of names. SMITH, SMITHE, SMITHY, may all become SNT or some other stabilized form.

Typically, the more characters which are removed or stabilized by an algorithm, the more error and variation is overcome. What is also true is that the more aggressive the stabilization, the worse the skew and hence the worse the search performance.

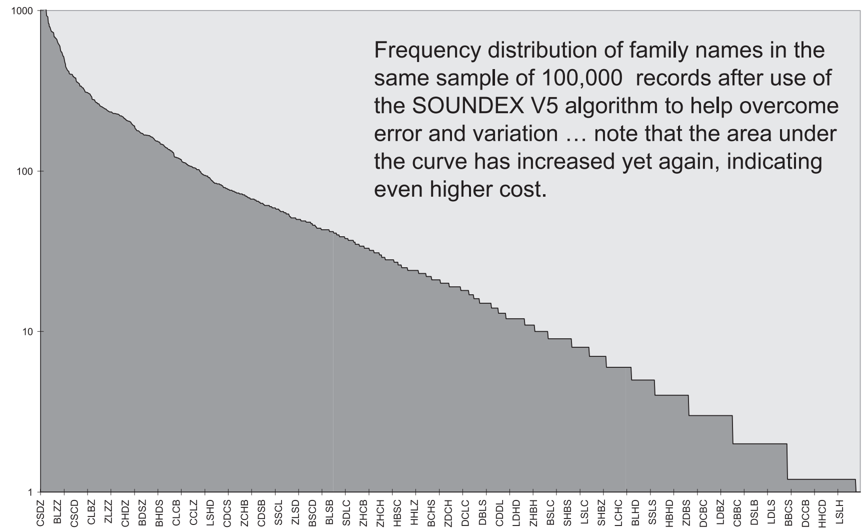
When evaluating Stabilization Algorithms, it is vital to keep the quality/ performance balance in perspective and to match the algorithm to the needs of the search application.

The following diagrams illustrate the effect on frequency distribution of three popular Stabilization Algorithms. These can be compared to the skew diagram based on exact name on the preceding page.





The choice of stabilization algorithm both affects the amount and type of error that can be overcome, as well as the number of unwanted or noise records that will be discovered. Different populations and business problems need different choices.



Choosing Search Strategies

The safest way of finding a name match in a database is to first perform a search on an index built from name alone, thus building a candidate list of possible matches, and then to refine, rank or select the matches in that candidate list based on other identification data.

The more of the name used in the key, and the greater the number of keys built per name, the greater the variety of search strategies that can be supported.

A search strategy that uses the full name makes sense when the name is expected to be generally reliable, when a match is expected to be found, or when the search can be safely stopped at the first match. Such a narrow search is feasible in a low-risk application that is looking for data that is expected to be on file.

A high-risk search, or a search with or against poor quality data, must use wider searches and/or more aggressive stabilization algorithms to compensate for severe spelling errors and more sequence variations. Such an exhaustive search is frequently used to prove that data is not on file.

In large scale systems the choice and sophistication of the search strategy is consequential to both performance demands, risk of missing critical data, need to avoid duplication of data and the volume of data under indexing.

To illustrate the complexity; to prove that Geoffrey Norman Holloway is not on file requires that at least the system efficiently checks there are no Geoffrey Norman Holloway's, no Geoffrey Holloway's, no G. N Holloway's, no G. Holloway's and no Mr Holloway's. However, in a large-scale database it would be far too expensive if this meant that the records for Gregory Holloway, or Gerald Nathan Holloway or in the worst case all the records with a word like Holloway in them, were to be processed.

When you add permutations, missing words and concatenations to this process the design and choice of strategies becomes a very important issue.

The choice of search strategy should match the business needs of the search. The search strategy used for one set of data or one system may be very different from that used in another.

Impact of Risk on the Search Transaction

In many business systems the risk of missing a match must determine the scale of the search.

Compare the risk of missing:

- a bad credit record when lending $1,000,000
  - as opposed to $1,000
- a criminal history record for a serial murderer
  - as opposed to a petty thief
- a border alert record for a terrorist
  - as opposed to a visa overstayer
- a medical history record
  - as opposed to a prospect history record
- a dangerous material advice record
  - as opposed to a yellow pages entry

In fraud, criminal and alert data, the important high risk record will often be harder to find because the identity alteration becomes more devious and complex.

In data which is collected over long period's, the important record will be hard to find because time may have altered the identification in the search data.

With complex or locally entered foreign data, an important record will be hard to find because of its tendency to contain severe error.

High-risk searches must be thorough. With today's data volumes, thorough searching must use intelligent keys and search strategies to manage the volume and quality of records returned.

Even with intelligent keys and search strategies, being thorough necessarily increases the volume of candidates returned. Because of this, reliable matching must also be used to assist the user by refining and ranking the list, and in some systems actually by matching the record, based on all available identity data.

Lower risk searches can afford to be less thorough, and can take advantage of assumptions about the stability of the data to provide quick access.

If you value your business, don't trust the same strategy or scale of name search for transactions of different risk value. You may need to automate the choice of strategy relative to the transaction's risk. Index the critical data separately and more thoroughly than the non-critical.

The Critical Negative Search

Some examples of critical negative searches are the search of a fraud file or denied parties list in a high-risk financial transaction; top level security clearance for government; a border control search of a high-risk person alert list.

Typical characteristics of such searches are,

- the volume of records to be searched is relatively low compared to the volume of searches done
- the bulk of the search data is more reliable and has different characteristics than the file data
- the search needs to overcome the fact that in many cases, that are very critical to find, the identity will have been manipulated to try to defeat the search
- the need to find a match if one exists is critical

A critical negative search must also be able to find identities that have been deliberately manipulated to defeat the system while still retaining enough similarity to be explained as mistakes. It will need to succeed despite the country of origin of the identity.

To do this, the critical negative search must work harder and look deeper. It will also benefit from working more intelligently.

Quality and performance will improve the more that is known about patterns used to manipulate identity data. Quality will improve the more identification attributes are available for matching. Attributes with null values may need to be considered close to a match.

Because there will be more candidates on average returned from a search, maximizing the true matches and minimizing the false becomes harder. In many cases the computer system alone cannot make the choice "is this a match". The system's success is measured by how well it assists the user make this choice.

Balancing the Risk of Missing Things with the Consequence of Finding Too Much

A designer of a strong name search will understand both the risk of a missed match and its cost to the business. When designing name search applications, recognize that each data population to be searched may have different risk attributes and costs of failure.

A missed match can be due to human error, because the name search failed to find the record, or because the match was 'hidden' in the results set (due to the list being too large, or not in a useful sequence).

A name search which fails to find a candidate match either did not cater for some types of error and variation, or did not look wide enough.

The more error and variation that is overcome, and the wider the search, the greater the potential for finding more true matches. The reality is, finding more real matches increases the amount of work and the cost. It also increases the risk that more false matches will be presented.

The goal of a good name search process is to maximize the true matches while minimizing the false matches.

Even after the name search process has been tuned to provide this balance, there will always be the tendency to find more true matches at the expense of introducing more false matches.

In the final analysis, a well-informed decision should establish the cut-off point. If it is decided that no matches are to be missed within the power of the name search, then more human resources will be required to select the true matches from the false. If it is decided that human and machine resources take priority, then the name search can be tuned to deliver to that level.

One of the serious problems of finding too much for an operator to look at, is that the human operator themselves then make poor decisions.

Even good well-trained operators cease to be diligent if they are expected to be searching hour after hour, day after day.

With well designed automated matching it is possible to build systems that mimic the very best human operators looking at all the available data and making decisions that are significantly better than the average human operator can achieve.

Undermatching v's Overmatching

Before a designer or user can decide what to show in a search or matching application, it is imperative to understand whether it is best to Undermatch or Overmatch.

It comes down to which case causes more or less problems for the business.

If it is simply a case of reducing the cost of mailing by avoiding duplicates then undermatching is good. Yet if it was important to avoid annoying the recipient, then overmatching would be good.

If it is a matter of not letting a known terrorist into a country or on to a plane, then overmatching is essential and, as in all security systems, a necessary consequence will be that some innocent people get inconvenienced by the process.

In a statistical process the consequences of undermatching can not be measured, but experiments can be designed to measure the amount of overmatching in the results.

In all designs it is necessary to know whether one would rather miss things, or rather find some things you did not want to find.

Once one accepts that error and variation in the data is normal and unavoidable, then it is true that absolutely correct matching can not be achieved, and it becomes necessary to decide if the "maybe true" answers should be seen or hidden from view. This is a fundamental business decision.

Nick-name tables are necessary, but there is a trap

Tables of nick-names are an essential component of a name search engine. Whether the person's real birth name is Bill or William, data will occur using either form. A search for BILL must also return WILLIAM a search for WILLIAM must also return BILL, and similarly for the initials B & W.

This equivalence must be established by recognizing one form of the name and either converting it to the other (usually the nick-name to the formal name) prior to indexing and searching, or by building a separate key or search for each name, one containing BILL and the other WILLIAM.

The trap is that many nicknames form an overlapping structure that if used gathers many obviously irrelevant names, e.g. Tina = Christine = Chris = Christopher = Toffer; or Al = Albert = Bert = Herbert

Converting the names to a common form has the advantage that less keys are built and less searches done. The alternative of permuting the names at indexing or search time has the advantage that the search for such overlapping names can be more selective. A search for Tina can search for Tina, Christine and Chris only, and a search for Christopher can search for Christopher, Chris and Toffer only. Of course the search for Chris must find them all.

A well-designed combination of the two methods will provide the best mix of reliability and selectivity.

Discovering the missed matches

One of the greatest myths regarding name search systems is that they are successful simply because they find what was expected or is known to be on file.

To truly measure the success of a name search, one also needs to have a real exhaustive understanding of what matches have been missed.

In many organizations, missed matches are only discovered once they adversely affect the business, operation or system. Whilst this is often too late from a business viewpoint, such discoveries are useful input for improving the name search process.

Missed name matches can also be discovered from within the organization's data by finding existing duplicates based on attributes other than name (e.g. address and date of birth), or by using controllable over-matching. To be useful for tuning the name search, this requires expert users to review the missed matches now found and help establish rules to avoid missing these matches in future.

Whatever the method of discovering matches that otherwise would not have been found, the goal should be to create and maintain a set of model answers, based on both real data and expert user input, as a benchmark for the reliability of the name search process.

When it is critical to a business or system to absolutely avoid missing data, then it is critical to implement procedures and processes to discover real world cases and examples of what can be missed. Only then can systems be improved.

Testing Name Search Transactions

The performance and reliability of a name search transaction is dependent upon the size, makeup and frequency distribution of the name population being searched. It is also dependent on the makeup and commonality of the search name.

To achieve useful test results, the name search keys and process must be tuned for, and tested on, real production data and real volumes.

Unlike many business transactions, where a finite set of test data can be defined to test all of a transaction's functionality, there is no limit to the name variations which may need to be handled by a name search.

Defining a limited test file of names is not good enough,

- a customer search which tests successfully on the 500 record employee file, is no test of how it will perform on the 5 million record customer file.
- a search which finds TEST MICKY MOUSE, XXXXXXX XXXXXXXXXXXX or THIS IS A VERY LONG NAME FOR TESTING, is no test of whether it will find EYAL LEBEDINSKY, ABUL MOHD AZIZ RAMAN or BILLY SAY LIM HO.
- a test search which uses the full name as search criteria is no good if the user ultimately only has a surname and an initial to search with.

Name searches should be tested, or the results evaluated, by expert users who can feedback reliable information to the designer.

It is not enough for a user to test only with the difficult cases not found by the old system. Tests should be carried out on more common names to ensure the search finds them as well and does not return too many.

Match thresholds should be set relatively low during testing to assist the discovery of matches which otherwise would be missed. This is essential for the process of tuning of the matching logic.

A batch test of an on-line customer name search that uses as search criteria a file of new business transactions, or even the customer file itself, provides a valuable report for users to evaluate the reliability of the search.

Because the system resource usage of the name search transaction is higher than most business transactions, it is vital that the expected volume and concurrency of searches be factored into any capacity planning.

Testing Batch Matching Processes

The batch search and matching process is best viewed as a series of on-line transactions   its input and output being file based rather than screen based.

The reliability of the batch process can be tested in much the same way as for the on-line transaction……. use production search and file data, set the match threshold low, print the ranked results to a report for human evaluation.

The match threshold for a production batch process is important to get right if matches are to be automatically processed further by the system. Set the final production threshold only after careful user acceptance testing.

Being only a single user, the performance, or run-time, of the batch process is simpler to test. Like the on-line transaction, it should use production volumes. Take advantage of non-critical processors for testing, provided that CPU and I/O time can be extrapolated back to the production environment.

As the on-line search is a two step process (retrieve candidates from a database then match them) so is the batch process. Optimization of the batch process (e.g. using sequential or flat files instead of database processing) should not be allowed to diminish this functionality.

Chapter

6

Parsing, Standardization & Cleaning

Systems must be designed to find and match data well regardless of its shape or order, regardless of its country of origin, and without the need for detailed parsing and cleaning of the components.

The Cleaning Argument

Making any substantial change to the identity data stored in a system, without also keeping its original form, is counter-productive to future needs for that data.

Using cleaned identity data alone for search and matching does not yield the highest reliability.

Storing identity data in essentially the same form as it has been supplied by the identity owner is a safe decision. For business functions that require it, storing an additional 'cleaned' copy of that data or cleaning the data specifically as part of the business function, is good design.

Matching systems that rely on data being cleaned prior to developing match keys suffer from the fact that a percentage of names and addresses fail the cleaning process (can either not be cleaned, or are cleaned incorrectly and destructively).

The problem is, the definition of ‘cleaning’ as applied to identity data is subjective and prone to error,

- If a name is input to a system as “MIKE SMITH”, should that name be cleaned to “MICHAEL SMITH” prior to storage, or do the person’s identity documents say “MIKE SMITH”?
- If the name was input as “SMITH MICHAEL”, should cleaning reverse the order, or was the order correct, if rare?
- If an address is input as “40 MARINA VIEW ST HUBETS” should “ST” be cleaned to “STREET”, or could it be “SAINT”?

These problems are amplified when international data is present.

Cleaning data prior to its entry into a Data Warehouse, in the interests of cross-system standardization, is only safe when no assumptions need to be made about the data values.

Once the original value is lost, there may be no history from which to reconstruct it if the decision was wrong.

In some data populations, where the rules for cleaning are well defined, the highest reliability from a match process will be achieved when the process can utilize both the original and the cleaned data; in other populations, where the reliability of the cleaning process is questionable, working with the original data alone will be the most effective.

The Real Value of Postal Address Standardization

Postal address standardization takes an address from the real world and attempts to validate all or part of it against Post Office rules. This sometimes results in a change to the real world address, or at least the assignment of some type of postal sort code.

A minimum result is that usually the city, state and zip code conform to the Post Office rules and are valid in respect of each other (a city with this name does exist in this state and has this zip code).

The real benefit of postal address standardization is the creation of efficiently deliverable mail by allowing bulk mailings to be sorted according to postal routes.

What Postal Address Standardization does not guarantee is that the mail is going to the intended addressee. This can be a result of poor matching against the rules and subsequent data corruption. The financial benefit of doing the mailing may, however, more than compensate for such mistakes.

Postal Standardization also can not guarantee that the addressee will appreciate a change in their address.

The real danger to the business is if the organization keeps this enhanced address as the default in preference to the real world address. The future ability to match to the address is then dependent on subsequent reports of it conforming to the same rules.

Avoid standardizing the customer's default address as this may adversely affect future matching. Store a Standardized Postal Address as a separate entity, or use Postal Address Standardization only as part of the output process that creates the mailing file.

The Value/Weakness of Parsing

Parsing of names and addresses analyzes them in an attempt to identify and attribute each token (initial, word or code).

Parsing relies on rules about token position, format and context. Punctuation and structure, if available, can assist. For some attributes, dictionaries are helpful.

The reliability of parsing is weakened because,

- names and addresses can be, and are, successfully used by people without following the rules;
- the rules are sensitive to spelling error;
- the rules differ from country to country;
- there is often ambiguity in the tokens;
- naming dictionaries are incomplete.

If the goal of parsing is simply to satisfy theoretical need, there is no direct benefit to the user of the data. The best format for real world usage of names and addresses is in the form of an addressee on an envelope.

Search and matching systems do not need to rely on parsing to build search keys or match codes there are more reliable methods. Critical search systems should never rely on parsing.

On the other hand, gross parsing of addresses, e.g. splitting an address into a 'fine' component (e.g. parts up to but not including town name) and a 'coarse' component (e.g. from town name to end), can be useful by reducing the noise returned in a search.

Selective parsing is also a viable solution for a number of less critical business functions—e.g. analyzing a name to discover the most useful word to use in letter personalization—analyzing an address to discover candidate town names for searching a reference table and applying statistics.

Chapter

7

Customer Identification Systems

One of the common applications of name search and matching applications is in Customer Identification Systems.This chapter looks at some of the issues facing the designers of such systems.

Consolidation of Customer Data

It has long been known that organizations must first consolidate customer, account or policy data from disparate systems prior to having a Customer Identification system that will benefit the entire organization, rather than just one business unit. CRM and Data Warehousing merely added some weight to this idea and produced some useful processes, however, they did not invent it.

The challenge for customer matching systems is the degree to which they can understand and overcome the error and variation that naturally exists in and between the source systems' data.

The measure of success to the organization is, in the end, how thorough and reliable are the relationships that have been created between the disparate customer records, not necessarily how 'correct' a cleaning process says a piece of data is.

When the goal is to consolidate data from more than one country or region, considerable care and thought must be given to how that data will be stored and later used. The costs and benefits of partitioning, multiple character sets and Unicode must be seriously considered, as must the issues of where search data will originate from and what form the search results need to be in to satisfy the business need.

What data to use for Customer Look-up

Customer look-up is expected to be both quick and accurate.

In some systems, the most frequently type of search will use an id-number, which is ideal for fast an often accurate retrieval. In other systems identity numbers are just not available or from a customer relationship point of view, the business chooses not to use them.

When an id-number is not available, the search will need to be driven by some other piece of identifying data. One of the challenges for the system designer is to decide which attribute or attributes are the best to use for this identity search.

Given a choice of name, birth date, telephone numbers or an address, how does one determine the best?

In an ideal world, one would try combinations of each attribute over a period of time and measure the system's results and the business benefits. In the real world, the decision often has to be made without empirical evidence.

Because dates suffer from the fact that a valid variation in any component creates a completely different but valid date, a search driven by a date is going to fail when one or more of the components are wrong.

Except where property addresses are the foundation of 'customer' (e.g. electricity and water companies), then addresses suffer from the fact that customers move regularly. A search driven by address is therefore going to fail when an address change has not been notified to the system.

Except when telephone numbers are the foundation of 'customer' (e.g. telephone companies or utility and emergency services), then telephone numbers suffer from the fact that customers move and change them, use home, work, mobile and public numbers. A search driven by telephone numbers is therefore going to fail when the number has not previously been notified to the system. And like dates, errors in the number, or variations in format make indexing with such numbers quite unproductive.

Names avoid the pitfalls of dates, phone numbers and addresses. Unlike dates or telephone numbers, if a character in a name is different, then it still has a good chance of being identified because systems can compensate for variation and error in names. And unlike addresses and phone numbers, names tend to remain more stable over time.

While names are clearly the most reliable piece of identity data to drive a customer search (other than an Id Number), the improvement in performance of processors and databases makes multi-level searches now a real possibility for lifting the quality of results. For example, a customer search that has name and address available to it can effectively perform both searches, even without the end-user being aware, and merge the results. Of course, even given the power of the hardware and database, to perform well in real-time such searches must use smart indexing and search strategies.

Use of Full Name in the Customer Search

An important characteristic of a customer name search transaction is that the average customer actually wants to be identified and will provide a full name when requested.

In the majority of cases, that name will be given correctly and will match the data on file. If the search takes too long however, both the customer and the system resource manager will generally complain.

Assuming the tuning of the system and database is addressed, the response time of a name search is dependent upon the commonality of the name, the volume of data on file, the richness of the file name and the design of the keys.

If 1% of the customer data is about SMITH, a key built from family name alone in a database of 1,000,000 records could return 10,000 records for the SMITH search. A key built from family name + initial might reduce that volume to 500 records, but that is still too many. In addition, 1% of the customer searches will probably be about SMITH and so the problem gets worse.

If the customer take-on system only captured family name, or family name and initial, then these difficult to use results are the best one could expect.

Provided the customer take-on system captures the full name, and given that we are expecting the average customer to provide their full name for future access, the name search should be able to take advantage of this to search a much narrower set of records.

This requires the operator to understand that using the full name will improve the response time. Such a system must also allow the widening of the search in case the match could not be found at the initial full level of detail.

Responsibilities of the Customer Take-on Transaction

Modern customer systems generally have access to complete person details, to large amounts of data storage and to application environments that accept variable size fields.

There is no reason why these systems should ever discard or truncate data as they did in the past.

One major responsibility of these systems is therefore to capture as much data about the person as possible within the boundaries of privacy laws and good customer relations.

A customer take-on application also has the responsibility of verifying the integrity of the person's details. This involves all kinds of edit checking, and at least a check to see if the person is already known to the customer system.

It may also be in the organization's interest to check other data sources for such information as:

- has this person applied before and been rejected
- does the customer have a poor credit history
- has the person been linked with fraud
- is the person on a identity watch/alert list

The most reliable piece of information to use to perform such searches is the person's name. It is generally the most stable, and can sustain the most variation without losing its essential identity.

The type of name search performed on each data should be allowed to differ due to the varying risk associated with missing a match.

Using the other identifying person data, such as birth date and address for confirmation (but not in search keys) these searches should be able to return a short list of highly likely candidate matches.

The Customer Take-on Transaction and Duplication

When a Customer Take-on System cannot find a match, there is a good chance that the operator will NOT perform any further searches, and simply add the 'new' customer as a new record.

Even when the system finds an existing record, if that record is not identical or not easily visible in the list a new record will often be added.

The important consequence of missing a match is not necessarily the duplication in itself, or the extra disk space taken up by duplicates, or the increase in candidates returned in future searches, but that the new customer record exists unlinked to the existing one. In future transactions it will be random as to which customer record will be used or updated.

Such unlinked duplication is a major risk to the integrity of the database. It is a risk to the business processes that expect to find only one record per customer, or at least to find all records relating to a customer together.

Duplication can be tolerated provided that the duplicate records are linked. Resolving duplication with merge/purge can cause data corruption and data loss.

Provided that duplicate records are linked, and systems are built to recognize the links, the decision to merge or purge duplicates becomes one of housekeeping rather than absolute necessity.

The real problem with duplication is when systems which use the data cannot resolve it, resulting in duplicate or unintended mail and even duplicate product being sent to customers, as well as a distorted view of the customer base.

Chapter

8

Multinational Systems

Foreign name and address data could be data sourced from foreign countries, local data from a different geographic or cultural background, or simply data of a type that has been previously unseen by your systems.

Such data is becoming more common in computer systems because of increasing multiculturalism, business globalization, electronic commerce, and because increasing amounts of identity data are being sold or shared in the market place. In some systems, such as Visa Determination and Border Control, foreign data is the norm.

A common problem when designing systems to handle foreign data arises in the design of the data capture format and rules.

Specific rules developed for local data will not work for the foreign data; individual designs and rules for each country or language is a maintenance overhead; and the idea of a superset of all possible fields and formats is too complex.

Requesting unformatted or loosely formatted name and address data is the best way of obtaining reliability, completeness and uniformity in global identity data. Asking for the data to be formatted according to strict rules is inviting assumptions and choices that can vary from person to person, country to country.

The different character set used to capture and store the data also poses another problem. It does not make sense to stabilize and lose that information if the data is to be used to reach the source again. Yet, to match such data it is necessary to ignore variation in the character forms.

The best approach is to request foreign data

and in its raw form, and to store it as such. Now, at least you have the best possible data available on the system.

Recognize that different business systems will want to use the data in different ways and leave it up to specialized software to overcome the problems associated with each business need. Don’t try to overcome these problems before the data is stored.

```
William Stuart Harison
   117- 2a Jacksen Rd.,
   East Hartford, CT 06987

Kwok Ki Ho (William)
   Block C, 4th Floor,
   Unit 7, 234 Wan Chai Road, Hong Kong

Mmd Farook Akbar
   Block A 27 Jalan
   Tuanku Abdul Raman,
   Kuala Lumpur

Augusto Frederico R. Schneider
   Aven. Maria C. de    Aguiar, 235   cj.
   32 São Paulo, SP   -   02593.001

Keser Geylani Abdulkadir
   Urt. Mahallesi
   Karaafat C.603/97
   S.No.186 Syhan/Adana
```

Field Design for Multinational Systems

Whether the multinational system is to operate in one country and accept data from multiple countries, or whether the system is to be deployed in multiple countries, the way that names and addresses are captured and stored is crucial to the reliability of future matching on that data.

Names and addresses from different countries have different structures, follow different rules and differ in average quality. (In Canada, it may be difficult to get a letter delivered without a postcode; In Hong Kong, almost no one uses the postcode).

A data model which assumes that the data for each country can be mapped into a detailed universal name and address format looks nice in the specifications, but will be costly to implement and generally unsuccessful in practice.

The universal format for a name is a single field holding all name parts. Simply make sure the field is big enough.

The universal format for an address is multiple lines, as written on an envelope. Simply make sure the field is big enough.

If the success of matching name and address data in your multinational system is important, do not trust match keys or matching logic which rely on the data being parsed, cleaned or formatted.

Use simple large single fields for name data, and a box of multiple lines as is used on an envelope for addresses.

A search and matching system that succeeds with the full unformatted name and unparsed address lines will be easier to implement, more flexible and ultimately give more reliable results.

Deployment of Multinational Systems

One goal of the designer of a system that is to be deployed multinationally is to reduce costs by minimizing customization, except where it is clearly necessary or benefits the user.

An example of where customization is often necessary is the language and font of the screens and reports.

An example of where customization is unnecessary is in the format of the fields used to capture and store name and address data. To simplify system expansion, these fields should be the same size and format for all countries and this format should be as universal and unrestrictive as possible.

In the construction of the database keys, search keys and matching logic, country or region level customization is essential.

The processes that build keys and perform matching should be able to succeed with unformatted or partially formatted data. However, while the original data should be captured in a free form, it is necessary to use key building, searching and matching algorithms that are tuned to each country or region of data.

If multiple character sets will be in use then character mapping algorithms, stabilization algorithms and tables for abbreviations, nicknames and other naming word rule bases will need to be externalized from the standard executable code. In some cases where multiple character sets and languages are in use in the one country, additional translation and/or transliteration rules may also be necessary.

These processes should be designed with a common interface such that implementing a new country requires only that new country-level modules and rule bases are plugged in.

Code pages, Character Sets and other Encoding Issues

This subject is not for the faint hearted; nothing in this area is as simple as we would all like it to be. Massive advances in character display technology, standards, tools and protocols have occurred over time. However the globalization of systems and databases has increased the frequency with which these standards are being mixed together.

Some examples of real world problems will suffice to raise the awareness of important issues.

It is true that accents on characters make them sound different but in most countries the error rate and variation in the use of accented characters is very high.

It is true that today's keyboard and code pages support accented forms, however many users still key the countries old conventions where two adjacent characters are used instead, or simply leave the special characters out.

We have found that databases in some countries suffer from non-standard versions of the local codepage standard. Fixing this still means that old data has different characters.

Moving data between tools sometimes converts characters without your knowledge. Some tools convert from EBCDIC to BCD and then back losing information. Some processes convert ASCII to EBCDIC and back inconsistently.

One terminal in a network set up with the wrong Code Page can cause database maintenance errors.

In a site in Chile we saw a large database where some terminals were using a USA English code page, others with a European Spanish code page, and others with a Latin America code page. This led to users continuously correcting and re-correcting the accented characters in a name and still each user was unable to see a correct form of the data. The net result is a very corrupt customer file.

DBCS encoding for Japan and China suffers from having several standards. This leads to increased complexity when sharing or comparing data from different sources.

The fact that people sharing data around the world can not read the same character sets as each other leads to names and addresses necessarily being recorded twice, once in a local form and also in an international form. In some cases this leads to the wrong form being used in the two fields, or even unrelated names being used in each field.

There are mixed protocols for handling foreign words, such as in Israel where sometimes Hebrew phonetic forms for a foreign name are used rather than the original Roman characters, or in Japan sometimes using Romanji and at other times using Katakana for a foreign word.

Different code pages and data entry conventions involving foreign data increase the complexity and error in identity data and this in turn increases the complexity of the algorithms needed to overcome the error and variation.

Unicode Issues

Unicode provides a technically more competent way of implementing international systems, and simplifies the storage, transfer and display of multi-lingual data. However, Unicode in itself does little to address the problems of searching and matching identity data.

- Unicode does not know that BILL is a form of WILLIAM, that ЛЁША is a form of ALEKSEI or that محمد is the Arabic form of MOHAMMED
- that 有 is essentially just 'noise' in a Chinese company name.
- that Ann Jakson could be a form of Anne Jackson-Brown

While it may be natural to think that Unicode can help unify data across countries and languages, Unicode does not help find and match identity data even within one language, let alone between languages.

Unicode can actually lead to an increase in variation of the identity data stored in a database if the data is allowed to be captured and stored in a variety of character sets.

Thus, the bilingual Greek/English data entry operator in England opening an account for a Greek-born British national (who has provided their Greek name on the application form), enters it in Greek because the system allows it. Worse, part or all of the name may even look like English (e.g. the name POZANA) and be stored as though it were an English name.

In the majority of systems, data entry should be restricted to the character set of the primary locale and converted to Unicode by the system. And it is essential that this locale information be kept and stored so that it is available for use by localized data matching algorithms.

Conversion to and from Unicode will require that it be done consistently. Conversion of old data to Unicode will still inherit all the error and variation in the old character forms. Users will still enter new data with the old character conventions, and of course continue to make mistakes.

Transliteration Realities

In most computer systems the term transliteration is used in the context of converting from a non-Latin alphabet to the Latin alphabet, or Romanization. In the real world, however, transliteration can occur between any two alphabets.

For example, a United States organization with offices in the US and Japan decides that all of its Japanese customer data should be captured in Japanese and in Romanized form to maintain a single language view of the corporate databases. A bank in Saudi Arabia captures customer data in Arabic for local needs, and in English to satisfy needs for inter-bank wire transfers and compliance regulations.

Transliteration may be done formally (conforming to a documented standard - although there will often be a number of standards used by different groups or organizations); or informally (by ordinary people in their normal day to day work, adding personal interpretations to the mapping choices and frequently changing the rules and making mistakes.)

Different formal transliteration standards and informal transliteration may co-exist in the same system/database, and result in significant variation in the transliterated form.

Transliteration also has an attribute of direction. Forward transliteration refers to transliteration from a name's original script to a target script (e.g. "Romanization" of an Arabic name from Arabic to English; "Arabicization" of an English name from English to Arabic.) Reverse transliteration refers to the transliteration of a name from its representation in a foreign script to its original script (e.g. "Romanization" of an Arabic name recorded in English back to Arabic; "Arabicization" of an English name recorded in Arabic back to English.)

In addition to data recorded in a local script, a system/database may contain data that has been the subject of any combinations of formal and informal, forward and reverse transliteration.

Transliteration and Data Matching

Transliteration can assist with data retrieval and data matching of identity data stored in foreign scripts, however, there are good and bad techniques.

Do not expect to achieve reliability and performance by transliterating multiple foreign scripts into a common character set and applying a localized matching algorithm to the result. There is too much conflict and compromise in the rules.

Search and matching on data from different countries and languages should be handled by algorithms tuned for each country/language.

Even a technique that attempts to detect language source in transliterated data to choose strategies and algorithms has inherent problems. How does one safely choose the language source for the name “Mohammed Smith” or “Charles Wong”?

If original script and/or informally transliterated data is available, do not discard it; such data provides an additional source of information useful for search and matching.

The real value of transliteration and transliterated data is when it is used in conjunction with the source language. A solution that indexes, searches and matches on all available forms, uses this inherent redundancy to multiply the opportunity for success.

Chapter 9

Identity Screening Systems

An important use for identity searching and matching in today's systems is the vital role it plays in identity screening.

Identity screening is used in a variety of systems including:

- Visa issue and Border control;
- Anti-Money Laundering (AML) Compliance and Know Your Customer (KYC) programs;
- Passenger screening;
- Pre-employment screening;
- Credit or Consumer screening;
- Marketing List suppression.

Characteristics of a Screening Application

Screening applications are about minimizing risk. The nature of the risk may be small or large…wasting the cost of mailing, damaging a relationship, doing bad business, doing catastrophic business, allowing illegal activity, putting someone's life in danger.

The screening data will often include both alert lists and 'cleared' lists (identities that should be expediently cleared). Making a false match to a cleared list is potentially as dangerous as missing a match in an alert list.

Regardless of the level of risk, there are some common elements of identity search in most screening applications.

- It involves a search where a "no match" is normal;
- A search where a "no match" is a good thing;
- A search where if an "alert" match could possibly be considered as true the system must report it; and/or deny the transaction or record from further processing;
- It must be designed and tested so that nothing relevant is ever missed.
- It must minimize false matches to avoid unnecessary and possibly expensive investigation, or false clearance in the case of a false match to a cleared list.

There are four points in a system where important screening needs to be done:

- To stop the transaction or raise an alert - involves a real-time screen of a transaction against an alert list;
- To discover historical matches (e.g. when better matching algorithms have been implemented), a periodic batch screen of alert list data should be performed against the database;
- At the time of adding new alert list entries, a screen of the new alert list entry against the database is required.
- Batch suppression of records from participating in a business process (e.g. marketing list suppression)

In addition, if a serious alert is raised, a common investigative requirement will be to find any other data in the organization's database(s) that could possibly be related to the identity that triggered the alert. Government and Industry investigation units need to be able do this across data sourced from multiple organizations. In this situation, because it is possible that the identity data involved is fraudulent or may have been manipulated, a thorough identity search must be used.

Identity Data in Screening Systems

Alert list data used by screening applications has different characteristics than typical customer or marketing data:

- It is generally of poorer quality (while some entries may be captured from official documents, many others are based on intelligence or third-party reports);
- An entry generally has fewer identifying attributes (only name may be present);
- The data is less complete (if an address attribute is present, the data may be missing or of little value; date of birth if present may be only an estimate of age);
- It will usually contain multi-national data;
- The multi-national data will be biased to a handful of countries;
- The skew in country/culture origin of the alert identities will be different than the skew in the transactions to be screened.

A specific problem in the banking industry is the need to screen financial transactions for AML Compliance. Such data (e.g. wire transfers, S.W.I.F.T., ATM etc) is often complex, only partially formatted and the identification details may only comprise a subset of the information.

In addition, it is common that, while the volume of alert/cleared list data is low, the volume of the transactions to be screened is high and be constrained by response time or throughput expectations.

Identity screening systems that use negative alert data require different strategies than other systems.

How do you Prove that you have not Missed Any Records?

There is no certification body for search and matching tools.

Testing these tools requires that, not only are you confident that all the data found is relevant, you must also be confident that very seldom is relevant data missed.

In systems that screen transactions against files of alert lists or other such negative data, the very normal and common expectation is that extremely few matches will be found. Designing testing strategies to prove that nothing relevant was missed, when the normal result is to not find anything, requires a lot of experience and skill in the testing of this class of system.

Many products that find “duplicates” in files have been purchased because of the high volume of duplicates discovered, when the critical criteria may still be “how many duplicates remain undiscovered!”

Failing to find anything is clearly a desirable and acceptable result of the process, but only if it’s true that nothing was missed.

By using software that has been used for long periods of time by organizations that have more critical needs and higher risk that your own, it is possible to be more confident that nothing is being missed.

The only way to be sure is by using software that allows controllable ‘overmatching’. If the software can be controlled such that overmatched results can be made visible and matches can be intelligently ranked in relative order of relevance of match, it is now possible to audit the quality of work. Such ‘overmatching’ is the only possible way to expose undiscovered ‘undermatching’.

The “False Hit” Problem

While not missing an important match is critical, false hits are also a primary concern in many screening systems. They are potentially a drain on investigator time and damaging to client relationships.

While controllable overmatching is essential for testing and audit, an operational identity screening system must be capable of minimizing the false matches.

In doing this it will need at times to cope with single word names, greater than normal noise, severe spelling errors, missing supporting data, foreign names, foreign character sets, fraudulent manipulation and more.

Such a system should be capable of finding matches such as:

```
SEARCH:     TONY DONG-SUNG GYUNG
ALERT:      KYEONG, ANTHONY

SEARCH:     INVOICE NO V-8021~TOSONI/CHAN-SHEI HAN
ALERT:      SHEIHAN

SEARCH:     ABDULLAH ABDULAZIZ ABDULLAH AL MUSA
ALERT:      ABD A/AZIZ A. ALMOUSA
```

While avoiding false matches such as:

```
SEARCH :    HERR FRANCOIS RIENERT / IM BUEHL 181
ALERT:      HERRI BAHALUNA

SEARCH:     FIDUCIARY BANK INTERNATIONAL OF NY
ALERT:      BANAKAAT-JORDAN INTERNATIONAL INC.
```

Chapter

10

Fraud & Intelligence Systems

In data used by Law Enforcement, Intelligence, Fraud and Security systems there is a growing need to support better reliability and availability, more data integration, increasingly diverse data sources and larger volumes of data.

Computer systems must make sure that the highly valuable data that is stored in these systems can in fact be found, despite its error and variation.

Similarly the value of the high-end tools of criminal and fraud investigation that provide 'link analysis', 'data clustering', or 'visualization' can be significantly improved if they make use of the very best search and matching algorithms.

Identity data in Fraud & Intelligence Systems

Many aspects of Fraud, Audit, Enforcement, Prevention and Investigation systems depend upon data about the names, addresses and other identification attributes of people and organizations.

All such identification data suffers from unavoidable variation and error. Often the data is out of date or incomplete. Often the entity committing the fraud or perpetrating the crime is in fact trying to defeat existing matching algorithms, by subjecting the identification data to deliberate, abnormal or extreme variation.

In systems which support intelligence and investigation work, databases of potentially relevant incidents and known perpetrators are maintained such that suspicious activity or new incidents can be linked or matched against them, or new patterns discovered.

Such databases require sophisticated indexing and search techniques that cope well with poor quality data, and provide timely and accurate results.

What Search Strategy to Use

Some solutions to the searching and matching requirements of such systems require skilled investigators who know when and how to vary a search or change the search data to cause the system to work more successfully. Boolean based and wild-card searches are an example of these.

A far better solution uses automated search strategies that satisfy all permutations and variations of the search…the real solution needs to be designed to find all the candidates regardless of the way the search data was entered, regardless of the quality of the data stored in the database, and regardless of the experience of the user.

Such search strategies must of course provide real-time searching of all name and identity data. On-line usage must satisfy the officer’s or investigator’s need for fast response without any loss of quality of search.

While diligent investigators can use sophisticated search tools well, it is not possible for the average user to spend day after day simply browsing historical data and do a good job selecting candidate matches; even the diligent user can get ineffectual at the job if it is a continuous activity.

To better automate the searching, matching and linking process, it is necessary that computer systems are designed to ‘mimic’ the very best users when choosing amongst the possible matches.

In the same way as human operators use names, addresses, dates, identity numbers and other data, the system must be able to use matching algorithms that effectively rank, score or eliminate the candidates.

How Well do these Systems have to Match?

When your CIS, CRM, Campaign System, or Call Center system fails to find a customer record that exist, you have an unhappy customer, or a lost opportunity to make profit. In this case, failing to find records that are present has a relatively small penalty.

Software that is good enough for “Duplicate Discovery” in marketing systems, or data warehousing systems will frequently leave undiscovered duplicates in the system the penalty is small enough for organizations to tolerate some failure.

When an insurance company fails to find out that it is doing business with a known perpetrator of fraud; when a Government welfare agency fails to discover that an address has been used for multiple fraudulent welfare applications; when a police officer fails to find out that the person in the car he/she just stopped is a serious threat, the penalties are likely to be large.

Chapter

11

Marketing Systems

Another common application of name and address search and matching applications is in marketing related systems.

Different Uses of Names and Addresses in Marketing Systems

Marketing systems use the names and addresses of people and contacts in a variety of ways.

- In matching and deduplication applications. For example, to dedupe a prospect list against itself; to dedupe a new prospect list against customer data, existing prospect data, fraud data or 'do not mail' and other suppression data.
- To reach prospects via direct mailings.
- To achieve cheap mailing rates by using Post Office preferred addressing.
- In scripts for telemarketing campaigns.
- To personalize letters, address labels and other marketing collateral to support a 'friendly relationship'
- In campaign preparation. For example to group prospects by household or location.
- To match incoming phone calls against campaign files.
- To support statistical analysis of campaigns. For example, to reconcile new customers by location against prior geographically based marketing campaigns.

Conflicting Needs of Name and Address Data

Marketing systems have conflicting needs in the way that name and address data is captured, stored and used.

In many marketing systems, this conflict has not been recognized, leading to a bias in one area and a less than satisfactory solution in another.

For example, the address most useful for reaching the prospect or customer and fostering a good relationship is the one the prospect or customer provides; the address most useful for achieving a cheap mailing rate is the one the Post Office provides.

Data that is parsed and scrubbed as it is captured into a system to support postal enhancement and personalization should not be relied upon for the development of a match-code for matching and on-line enquiry.

Incorrect parsing destroys valuable data. Original data must be retained to support high-quality matching.

Even if a match-code process that relies on cleaned and formatted data is used for the marketing system, it should never be used for systems where missing a match is critical (e.g. fraud, audit and intelligence systems).

Chapter 12

Summary

Fundamental Characteristics and Components of Strong Name Search and Matching Systems

The fields used for keys and matching should be raw data fields like “name” or “address” rather than a list of separated specific elements.

Original real world data should be input without preprocessing. Databases must retain this original data.

The search engine and matching algorithms should be able to search and match as well as the best human experts in the organization can.

Rule based and procedural programming will be used to:

- control an editing phase to recognize items that are case and punctuation dependent, such as certain common company name abbreviations. e.g. s/a , c/o ;
- overcome character representation variation, such as casing, accents, delimiters, punctuation etc.;
- recognize and ignore ‘noise’ words;
- recognize and treat as identical the common synonyms, abbreviations, translations, nicknames, ethnic and anglicized forms of words;
- stabilization algorithms to overcome error and variation in unrecognized words;
- build multiple keys or signatures from the transformed and stabilized data.

Philosophy and Convictions about Name Search and Matching

There is no such thing as an invalid name or address. Search and Matching must be possible on data that can not be understood, parsed or scrubbed.

Systems must be designed to work with whatever data they can get, rather than the mythical data that the designers would like to have.

Raw original real-world data contains more identification data and quality than enhanced, scrubbed and parsed data.

Data enhancement and scrubbing should only be used for reporting purposes not search, matching or identification, because any failure or error during scrubbing or enhancement of the data will reduce the quality of all future search, matching and identification.

The maximum quality that the data can support should be achievable despite performance and cost.

Tools should not restrict the quality. The application designer must, however, be able to tune the balance between quality and performance for specific transaction types and purposes.

As it is true that business risk varies with transaction values, so it must be possible to vary the cost/performance ratio of name search transactions, to match the risk associated with the transaction.

The quality, uniformity and reliability of name and address data is declining with the era of electronic transactions, global business and personal data entry.

Whilst poor quality data may limit the value of data, all systems should be able to process and match data regardless of its poor quality.

All customer and marketing databases will contain a percentage of data that is from 'foreign' marketplaces.

Tools must work well regardless of the country of origin and language of the data and our tools must insulate the applications system developer from the differences between country and language, when it comes to name and address search and matching.

Tools should not demand significant local knowledge or be dependent upon the maintenance of databases of current postal address information. The ongoing daily change in this data creates a continual burden and weakness in the users business system.

To get good response in name search you must denormalize and maintain a copy of the relevant name search and matching data in a file or table optimized solely for name search and matching.

This table will contain an entry per ‘fuzzy key’ together with secondary identification data used to make the final choice. To optimize access to this table or file it will be physically ordered on the fuzzy key that will not naturally be a unique key.