Identity Resolution Explained

Presented By:

William Indest

Bruce Hamman

ORACLE®

Executive Summary

Businesses around the world must be able to uniquely identify their customers. The Oracle Master Data Management (MDM) solution includes capabilities for identity resolution, a process that can be interactive or batch.

In "interactive mode", when a user enters a person's name into an "MDM Aware" application, a match request is sent to your master repository and returns one of the following: 1) no matches found; 2) one match found; 3) several matches found. Let's say several candidate matches are returned to your end user, they then select the candidate with the highest match score.

The person's profile information is retrieved and populates the appropriate fields in the application allowing the end user to be more productive since profile fields are pre-populated, data duplication is avoided and a countless amount of cost savings realized.

"This paper will show you how a name match with a high degree of certainty is accomplished. "

When the end user's application is "MDM Aware", it has the capability to send certain attributes to a central master repository, perform an intelligent match and retrieve any found candidates in real-time assigning a confidence score to each record returned.

How does this matching process work? How do we know if the match is a good one? What is going on inside the matching black box?

This intelligent match is based on searching a typical array of fields, name, address, social security number, drivers license number and can be fine tuned for other attributes or keys, all specific to your requirements and needs. This process guarantees a solid match if the person exists in your master information repository.

Setting up this intelligent match capability involves a non-intuitive concatenation and re-ordering of all of the names of a person combined with algorithms specific to verifying a person's address(es) and creating sub-sets of the name information so that the search executes quickly in a small search domain.

In addition to a robust matching process, the Oracle MDM solution allows you to set thresholds to determine what level of confidence is acceptable for either auto-match/merge or create a new record to oc-

cur when loading records. A match score above the upper threshold means the input search record exists and can be merged to create another Best Version. A match score below the lower threshold means the creation of a new person.

If a match score falls between the upper and lower threshold, then the input search record and candidate matches are presented for manual review or human intervention. This capability allows for flexibility to fine-tune your data governance process. This paper will show you how a name match with a high degree of certainty is accomplished.

Introduction

In his book, The Back of the Napkin, Dan Roam states that all problems can be categorized in a basic set of six clumps; his six W's are: who, what, when, where, how and why. Master Data Management attempts to find out the who from multiple systems across a department, an agency, an enterprise, a state or even a country.

The process of identifying who a person really is from various information repositories is daunting to be sure but the tools used to solve this problem are mature. Herein we present an excellent way to address this challenge and obtain results that make the best version exactly that – your best determination of who a person is.

In many systems we need to find things that have been filed away using a Person's name, a Company's name, an Email address, a Place name, an Address and so forth. All such names are collections of words and numbers used to "label" or "identify" the original real world entities. In systems and databases we use such names to find many types of data recorded about the "entity" identified by the name or naming data.

" Names...are subject to a lot of variation and error. You cannot avoid this variation and error."

Here are a few facts about these names:

Names are generally not unique.

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

You cannot 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.

Furthermore data decay occurs constantly due to the dynamic nature of the human species.

(An excerpt from "The Math, Myth, & Magic: An Introduction to Identity Data Search-and-Match", Informatica, 2008).

Errors and Variation

Here are a few of the data errors that plague every system:

20 Common Errors and Variation

Variation or Error	Example
Sequence errors	Mark Douglas or Douglas Mark
Involuntary corrections	Browne – Brown
Concatenated names	Mary Anne, Maryanne
Nicknames and aliases	Chris – Christine, Christopher, Tina
Noise	Dashes, slashes, titles, full stops and apostrophes
Abbreviations	Wlm/William, Mfg/ Manufacturing
Truncations	Credit Suisse First Bost
Prefix/suffix errors	MacDonald/McDonald/Donald
Spelling & typing errors	Porter, Beht, K8
Transcription mistakes	Hannah, Hamah
Missing or extra tokens	George W Smith, George Smith, Smith
Foreign sourced data	Khader AL Ghamdi, Khadir A. AlGamdey
Unpredictable use of initials	John Alan Smith, J A Smith
Transposed characters	Johnson, Jhonson
Localization	Stanislav Milosovich – Stan Milo
Inaccurate dates	12/10/1915, 21/10/1951, 10121951, 00001951
Transliteration differences	Gang, Kang, Kwang
Phonetic errors	Graeme – Graham

" All of these variations and errors can be addressed in a robust way."

Although it seems we have our work cut out for us, all of these variations and errors can be addressed in a robust way. Let's look at the tools available, examine what data quality means and the process involved and then process a record to better understand how Oracle's intelligent matching works.

Tools to Fix the Problem

The two tools used to address this identity problem are called Oracle Data Quality Matching Server and Oracle Data Quality Address Validation Server. These two tools are part of Oracle's Master Data Management application or the Oracle Customer Hub.

We are going to focus on the Matching and Address Validation servers and what processes are involved to determine the identity of a Person whose minimal set of attributes are first name, middle name, last name, SSN, street address, city, state and zip code. Let's first look at what data quality means. Figure 1 depicts the components of any Data Quality solution.

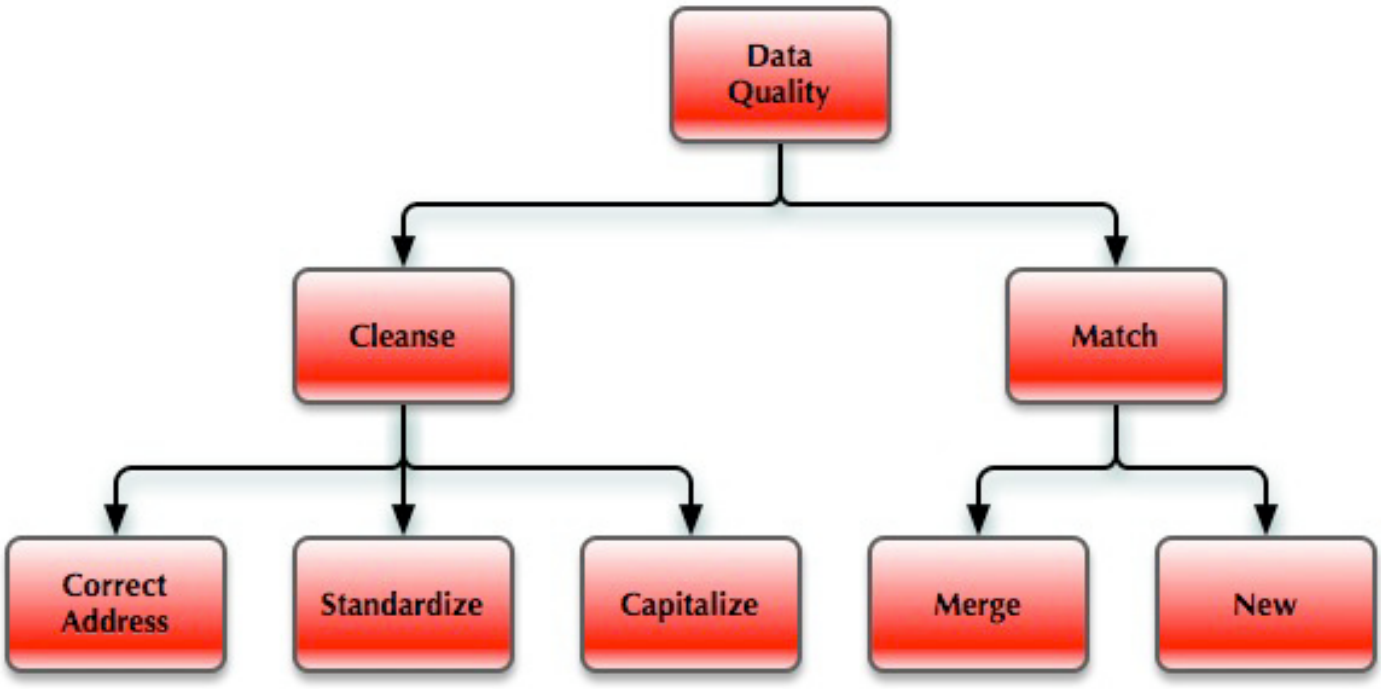


Figure 1. All of the pieces of the data quality solution, including matching.

There are two top-tier components to data quality, 1) Cleanse and 2) Match (or de-duplication). Cleansing involves correcting addresses, standardizing names and capitalizing appropriately. Matching involves determining if a Person exists and to either merge with the appropriate existing Person or to create an entirely new Person, your Best Version.

Person Data Management

The Person Data Management process involves loading the information into the MDM repository, cleansing and validating the addresses, then running the cleansed input records through the matching engine. The results from the match engine wind up in one of three buckets based upon your match thresholds.

" The results from the match engine wind up in one of three buckets..."

Assume you set your thresholds to a score of 95 for the Auto Merge bucket, 80 or less for the New Best Version bucket and anything in between 80 and 95 goes into the Manual review bucket. The process continues with the best version undergoing the survivorship rules. If publication is turned on then those subscribing systems are updated with the new information.

A graphical representation of the process may be found in Figure 2 below. Note that no information is destroyed in the construction of the best version and rollback to previous best versions is allowed. This is a key audit feature.

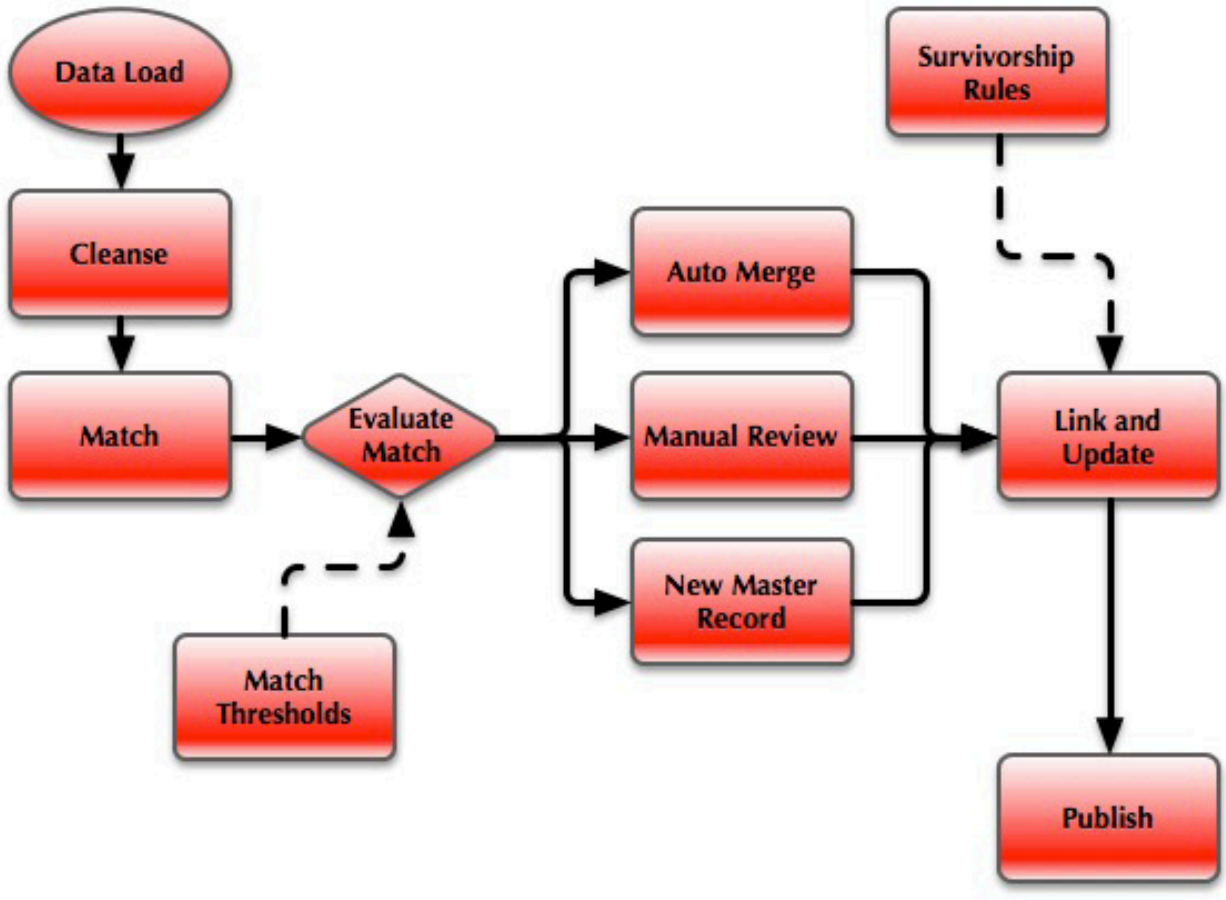


Figure 2. The Person Data Management process from beginning to end.

Address Validation

Let's focus on the Address Validation part of the process. Since address data can vary from country to country Oracle's Address Validation Server provides support for up to 5 lines of unstructured data. The Address Validation server will parse this data into the proper component fields based on the country tables and parsing rules. The server recognizes and separates non-address related data.

In this parsing step, valid address data is separated per country into it's proper fields; it does not validate that address against the countries' postal codes and parsing does not check whether the address fields constitute a valid address. Once the address data has been parsed into it's proper constituent fields it is validated against the proper country postal directories.

The validation process will attempt to correct errors if possible. In the US and Canada rules are applied to limit address changes. Address verification accuracy is controlled by the level of accuracy in the countries postal directory.

Here is a sample of how an address is parsed then validated.

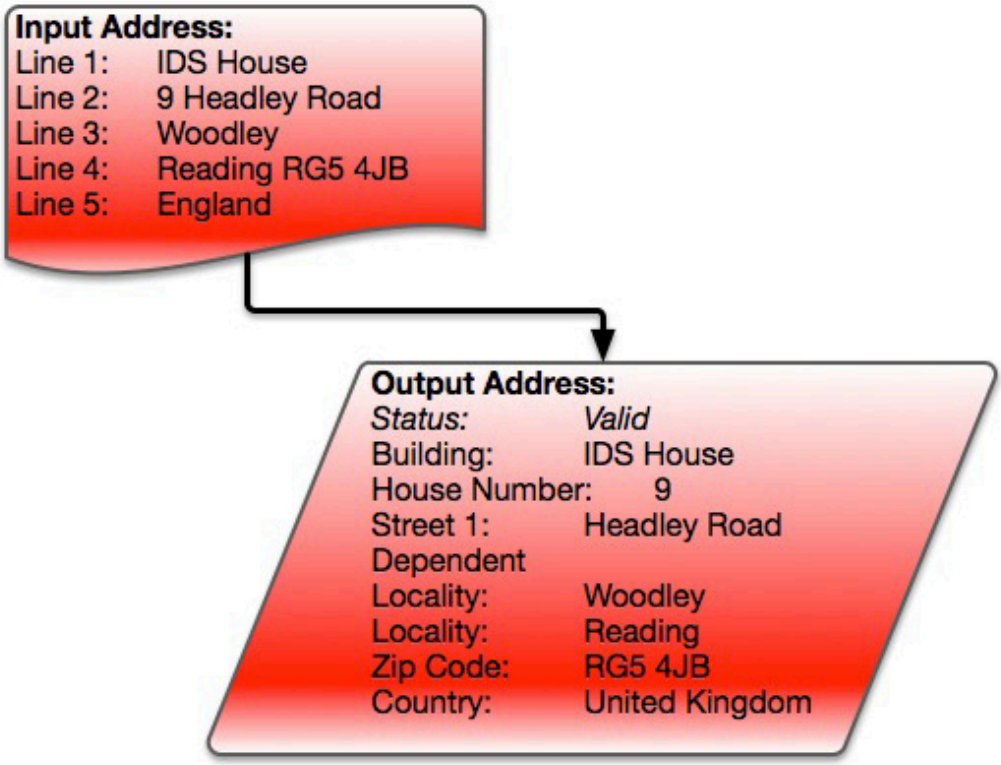


Figure 3. Input address parsed into a more meaningful output address.

Notice how the line 2 input has been parsed into a house number and street address and how the line 4 input is parsed into Locality and Postal Code. This is based on the country tables and parsing rules which know the required format for UK addresses.

The validation status indicates whether the address was valid, as it is here, and if not, whether it has been corrected and will indicate either: 1) corrected some error and variation in the input address; or, 2) corrected the postal information if the input address contains an erroneous or out-of-date code.

For uncorrected addresses, the validation status indicates the likelihood of mail being delivered successfully to the address and the validation process may generate a number of suggested addresses when the input address is ambiguous. Here are some validation examples:

The input address, "645 Maddison Avenue, New York, NY 10022", returns a status of "corrected", as the street name was slightly modified (Maddison was changed to Madison).

The input address, "1 Sesame Street, New York, NY, 10004", returns a status of "deliverability: small", as the street name wasn't found in the postal reference data.

Because of the various sources of postal data, the level of data for any given country may vary. As these postal directories change frequently they are maintained by Informatica and may be purchased separately from them.

To review, we have submitted the address fields to the Address Validation server to standardize the address fields. This is an important part of readying our Person record to be submitted to our Matching engine as the address fields are necessary for intelligent matching.

Generate Indices

At this point the address fields are standardized and validated. As the matching algorithms are usually set to look at the address fields we have minimized the number of indices we need to create for the address fields.

For example if we had 123 Elm Street and 123 Elm St. scattered across our repositories, we would create two indices to match so having one address, 123 Elm St., let's us create only one index. This optimization helps with the next step, matching.

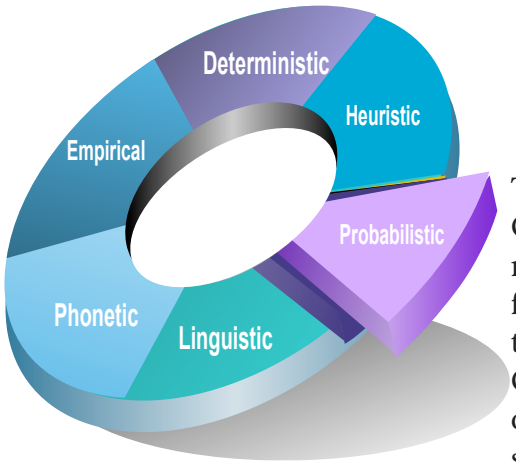
Once the data has been cleansed the record is sent to Oracle's Matching Server to be matched. In your initial set up, you define the fields you want to use for matching. These are your key fields. You are not limited in the number or type of fields you want to match upon. Oracle's Matching Server is not limited to a single search or matching strategy.

Oracle's Matching Server uses a two part process - the first step is selecting potential candidates and then executing matching algorithms against this candidate list; you have finite control over each step in this process.

The matching engine provides high quality search and matching on all types of name, address and identification data such as:

Person Names
Organization Names
Address elements
Dates
Telephone numbers
Social Security/Drivers License/Passport Numbers

A tremendous amount of support for highly flexible search strategies is available. The matching engine supports fuzzy logic to emulate a human expert's ability to match across many attributes. The matching engine employs smart indexing to cut through error and variation in identity data, including multi-country, multi-language information.



Matching Solutions

There are hundreds of algorithms that exist to solve this problem. Oracle's Matching Server uses several of the industry's leading matching solutions. This is considered a hybrid approach. The following six matching solutions are the most commonly used within the Data Quality/Identity Resolution industry and are also used by Oracle's Matching Server. By combining the best of each algorithm in our hybrid approach, we are able to provide the most comprehensive solution.

"Oracle's Matching Server uses a hybrid approach. It uses all of the industry's leading matching solutions."

A **Probabilistic** algorithm is based on statistical frequency analysis, and derives key values that are used to perform matching. This algorithm is not able to catch very common errors such as character transpositions.

A **Heuristic** algorithm operates on "rule of thumb" derived from experience that may be true. This algorithm cannot deal with data anomalies for which a rule is not present.

A **Deterministic** algorithm is one where the behavior can be predicted from the input. It is not able to resolve such common data anomalies as blank fields, transposition of characters, or abbreviations.

The **Empirical** algorithm is driven by data, based on experience or observation and can also reflect distinct cultural standards. It is dependent on dictionary/rule sets and cannot compensate for any "new" errors or variations.

The **Linguistic Language Recognition** algorithm is based on identifying what cultural background a given name comes from within a dictionary of names. This algorithm is not able to easily compensate for new names or hybrids. Valid variations could mean that incorrect rules are applied leading to missed matches. Such techniques fail, not just because new names are being invented daily, but also because valid variations in search data could cause name recognition algorithms to use incorrect name variations to perform the search.

Phonetic Algorithms create indices based on pronunciation. Most phonetic algorithms are based on English, and cannot compensate for non-English pronunciations. The results are unranked and unordered in relevance, and fail when even the first letter of a name is incorrect.

Create Search Nets

We now have an input record and we want to know if it exists or not in our master system. Initiating a search and matching each record against the entire customer database is performance prohibitive so we create a subset of the database which is called a search cast net to limit our search based upon a certain and specific search space.

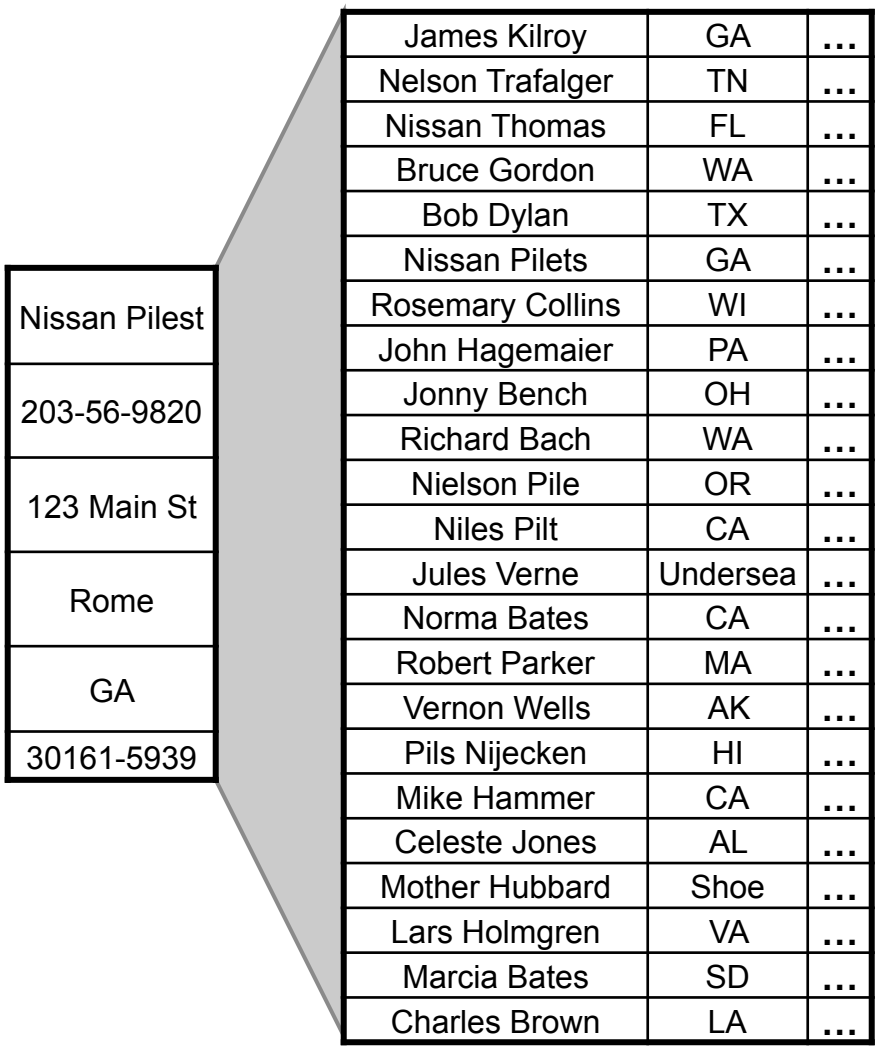


Figure 4. Nissan Pilest is submitted as a search request across the entire citizen database.

The task is to balance performance with accuracy so we identify and select only those records that have a possibility of matching and create a subset.

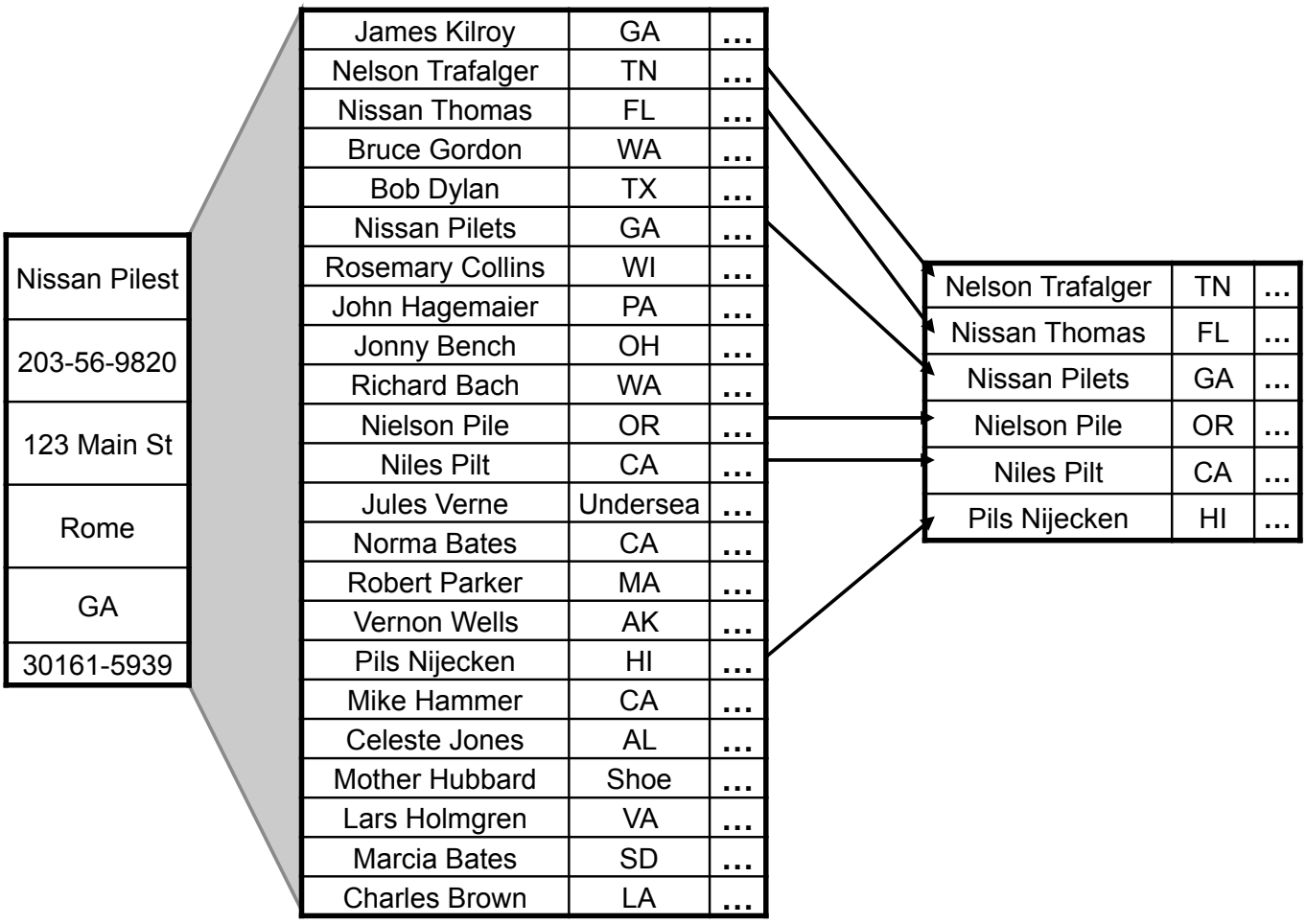


Figure 5. Creating a smaller subset of names to match to balance accuracy with performance when Nissan Pilest is submitted as a match request.

We think the best thing to do is to build keys/indices from the citizen file and use these keys in an intelligent manner to identify only those records that have a potential to match. This is how the Oracle matching server works best without sacrificing accuracy yet providing optimal performance.

Then we match only on this subset of records.

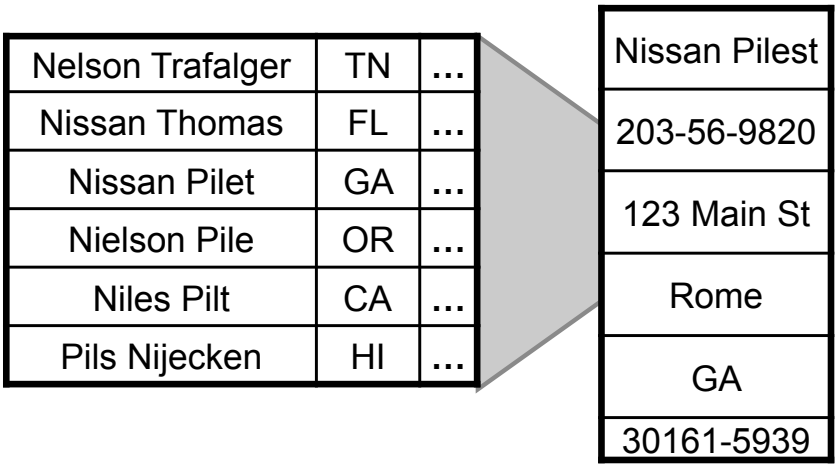


Figure 6. Input a search candidate, search the specific name sub-set and return a match, matches or none.

As part of your Master Data Management implementation you will define indices for each matching purpose described above. These indices are generated at the time a record is added or updated.

Oracle provides an out-of-the-box template for Person and Organization; however, you can create an unlimited number of indices. In the example below there are eight indices on the name, Ann Jackson Smith so...

From: Ann Jackson Smith we get:
SMITH + Ann + Jackson
ANN + Jackson + Smith
JACKSON + Ann + Smith
SMITH + Jackson + Ann
ANN + Smith + Jackson
JACKSON + Smith + Ann
ANNJACKSON + Smith
JACKSONSMITH + Ann

This is important because multiple indices help to eliminate false negatives as more records are selected for the candidate pool limiting transposed and concatenated word errors. Each word can have a Major or Minor position. This positioning helps determine the range of records selected as Major words are more important than Minor words for names. Major and Minor words are less important with Organization and Address fields.

Before we begin our search we can also modify the search specifications by editing the list of rules. An “Edit List” is used for each population of data such as: person names, company names, street addresses, or German person names, Italian person names, to address certain known characteristics of the data. The "Edit List" identifies nicknames, abbreviations, “noise” words, as well as phrases such as “trading as” that are used to identify compound names. (In a compound name, the Oracle Matching Server will automatically treat each part of the name individually, and build separate keys on each of the components.) Your "Edit List" may also be customized as appropriate for your population; this is an override capability.

Selection Criteria

Now that we have the custmoer file indexed and our field edit overrides in place we can turn our attention to the selection criteria. Remember from our earlier discussion about performance that we want to select a subset of records for each suspect record that is smaller than the total set of records.

We also want to keep these subsets to a limited size which is based on your performance requirements. In order to do this, Oracle Matching Server uses the commonness of names to partially determine the width of the net cast to select candidates. Oracle Matching Server generates these common name files on the commonness of names within your population set.

If a name is common such as John Smith then the search will be refined to limit the number of candidate records so that performance is enhanced. If a name is uncommon such as Nissan Geoffrey Pilets it will expand the search to enhance accuracy. This is of course can be modified for local variations within your citizen population.

You decide how comprehensive you want your search to be. You identify different sets of search candidates for different reasons:

A “Narrow” Search is a compromise on completeness in favor of speed.

A “Typical” Search is for most on-line and batch search transactions; this is a practical balance between quality and response time.

An “Exhaustive” Search is for high-risk systems, or to prove no valid matches exist; you obtain high quality, but you may extend the response time.

Oracle Matching Server creates search specifications based on the Major and Minor words created in the indices and based on the suspect record. Designating a narrow search will create the most specific search and the least records returned which sacrifices completeness and accuracy for speed.

As you can see from the simple graphic in Figure 7, a few records from the set of all Hiembaughs have been selected.

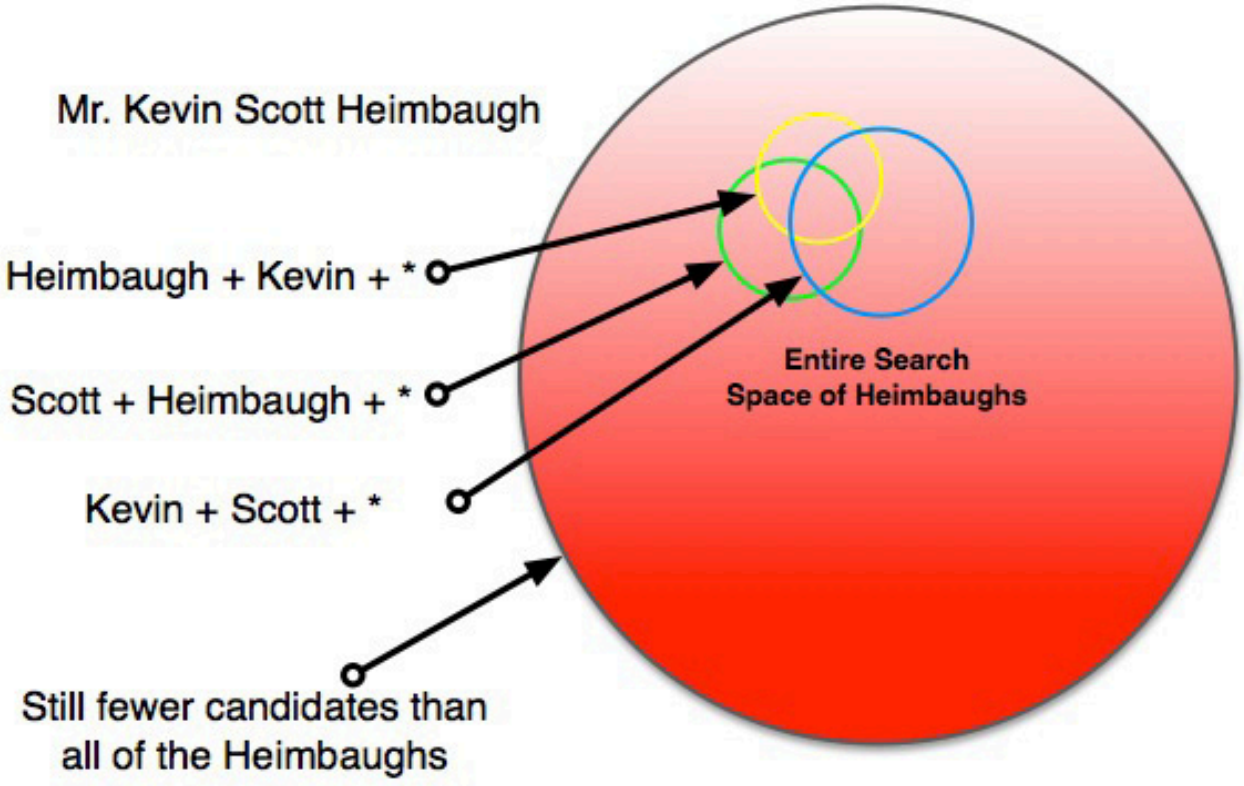


Figure 7. Narrow or Typical Search designations search less candidates but are better performing as not all of the Heimbaughs are searched, just a smaller portion. The asterisks indicate return everything in that position.

In summary, a **Typical** search designation will yield a less restrictive selection generating more results than a **Narrow** search. A **Typical** search strikes a balance between performance and accuracy and is used in most implementations. You can designate an **Exhaustive** search which will return all records containing the Major words. This sacrifices performance for the sake of accuracy.

Match Scoring: Development

Once we have our limited list of candidate records we need to compare each candidate row (record) with the suspect row (record). Our canonical example has Nissan with his SSN, City, State and Zip Code. We will examine different strategies on how to best create a match score.

Notice the different last Names and SSN. Remember from our indexing exercise that it is better to place the entire name in a single field so that the Oracle Matching Server can match regardless of name order. (This will not be the set of fields you would use for your particular implementation. This particular set of fields allows us to show the variables you can control in the matching process to generate a score.)

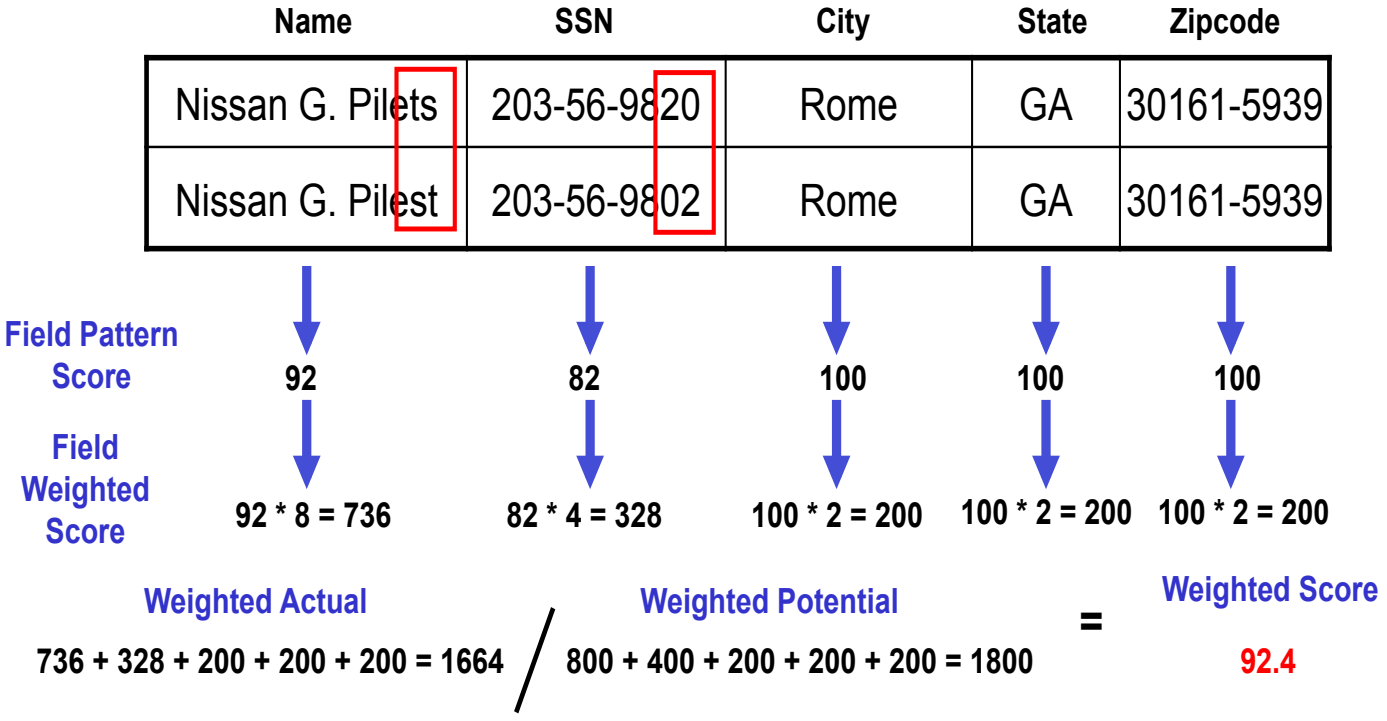


Figure 8. Match score simulation with Nissan Pilets and Nissan Pilest. Note the last two digits of the SSN are transposed.

The first step is to compare each field from the suspect record with it's counterpart in the candidate record. The Oracle Matching Server has more than 90 field compare functions. These functions make

decisions based on whether there are multiple words in the field, the field types, and other criteria to decide which algorithm to apply to each field.

For example, you would want to match dates differently from Names or Identifiers like Social Security Number. This is a process that can be modified based upon your particular requirements. The output is called the Field Pattern Score as shown in Figure 8.

The next step is to apply the matching rules you've setup. In this case we have simply added weights. Here we have given a weight of 8 to Name, 4 to SSN, and 2 to City, State, and Zip Code. These weights do not have to add up to any particular score as they are weights relative to each other.

In this case the weights make Name four times as important as City, State and Zip Code, and twice as important as SSN. Multiplying the Field Pattern score with the weights gets us the Field Weighted Score.

Summing the Field Weighted Scores and dividing it by the Sum of the Potential Weighted Scores yields the Weighted Score. In this case 92.4 is the result in red.

In the Person Data Management process (Figure 2) there are three buckets into which a match will fall - Create a new Best Version record, Auto Link this Source record with the Candidate Best Version record, or send it for manual review.

Which bucket the result falls into is determined by setting upper and lower thresholds within the process. Based on our thresholds of 80 and 95, this record would be sent to the Data Steward to review because it falls into the manual review bucket.

As a note, if there are multiple candidate records over the upper threshold, say both with scores of 98, you should send all of them to the Data Steward for review because the system cannot and should not auto-merge them.

Match Scoring: Missing Data

What happens when we have missing data? In this case we have a complete candidate record and an incomplete suspect record. Without modifying our match rules these two records would generate a match score of 70.2 which would create a new record for our suspect record.

As you can see this penalizes the match because fields are missing data. There are two potential ways to avoid penalizing a record when there is missing data.

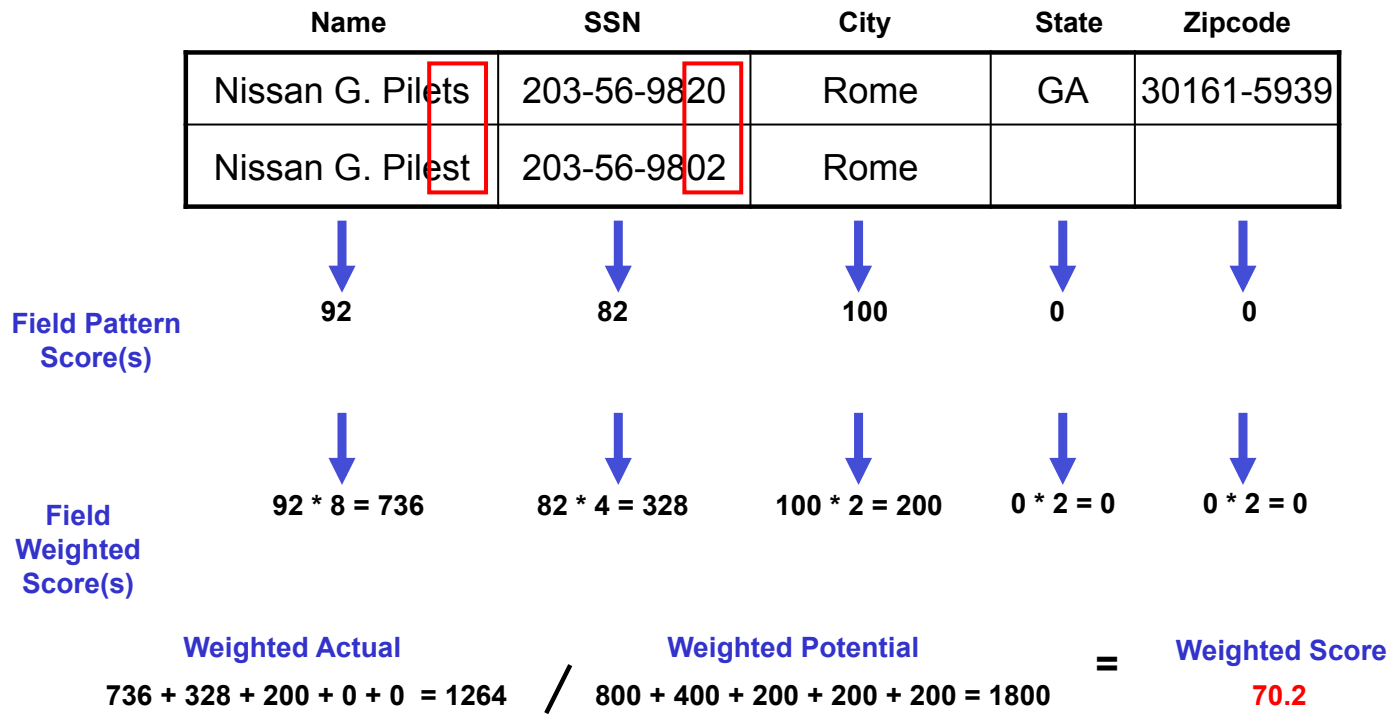


Figure 9. Match score simulation with Nissan Pilets and Nissan Pilest with missing state and zip code values.

The first is to designate a group of fields. In this case City, State, and Zip Code. Then apply the Best Group match parameter. This means that only the best score in that group will be considered as part of the match process. Notice this drops our Potential Weighted Score by 400. This raises the match score to 90.2. This is a more accurate score as it reflects only the values in the three fields you have.

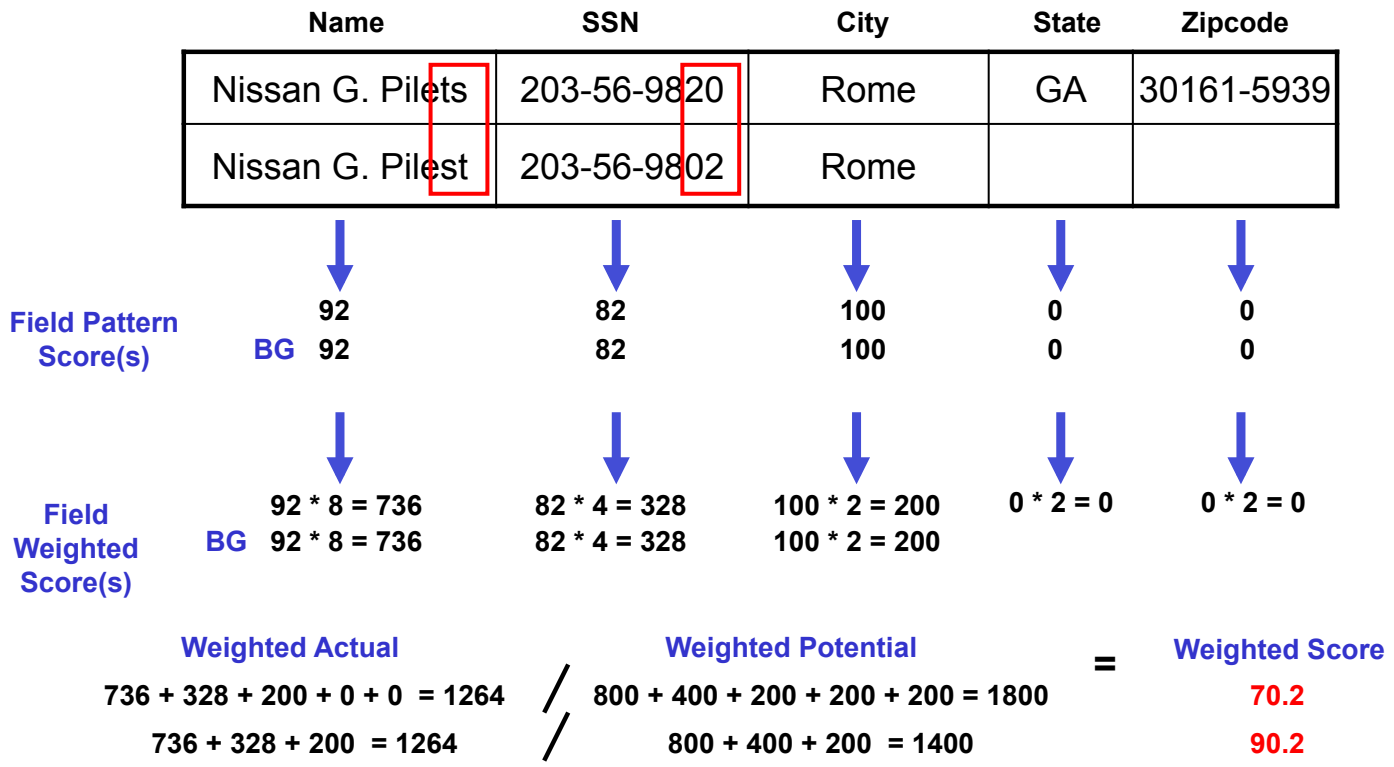


Figure 10. Match score simulation with Nissan Pilets and Nissan Pilest to show Best Grouping capability and outcomes.

Maybe we should not consider address at all unless two out of the three fields are available. In that case we designate the same group but apply the required group match parameter and indicate the number of fields that are required to be present within the group to be considered for the match process.

In this case it lowers both our potential and actual scores yielding a match score of 88.6. Now this may not seem that much different in this case from the BG (Best Group) score of 90.2 but if one of your thresholds were set at 90 then these two scores would generate different bucket placements. One might be auto-merged and the other could go to the Data Steward for review.

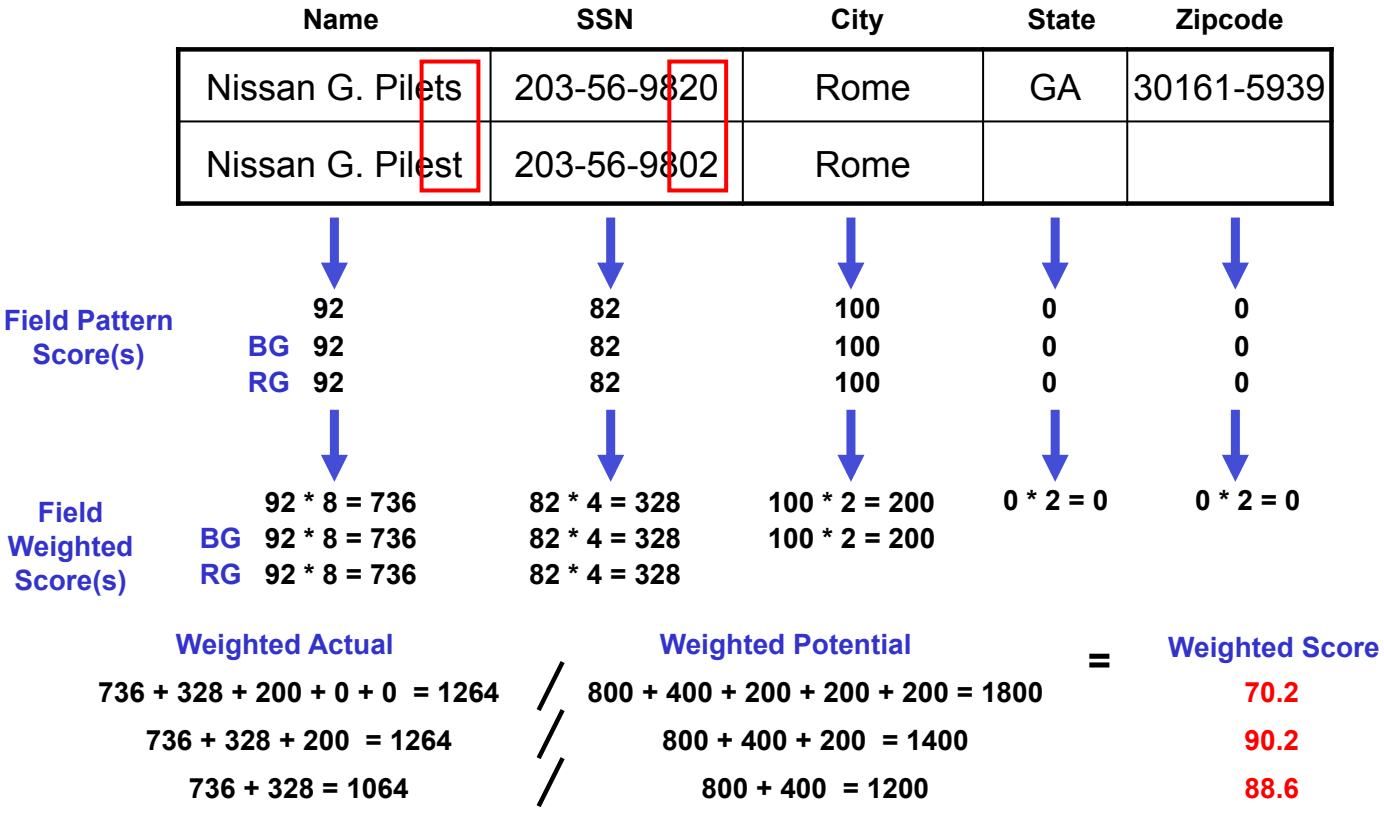


Figure 11. Match score simulation with Nissan Pilets and Nissan Pilest to show Best Group and Required Group capabilities.

Match, But Not Really

Now let's consider what happens when we have data that generates a match score but really is not a match. In this case our candidate record and our suspect record have significantly different SSN's.

In our previous examples our SSN field had a single transposed digit set. Now we have two digit sets that are different. How we handle this will significantly affect the final match score. If we do nothing, then the SSN field matches are calculated normally and we have a match score of 87.5. Fairly high for two people with different Social Security Numbers.

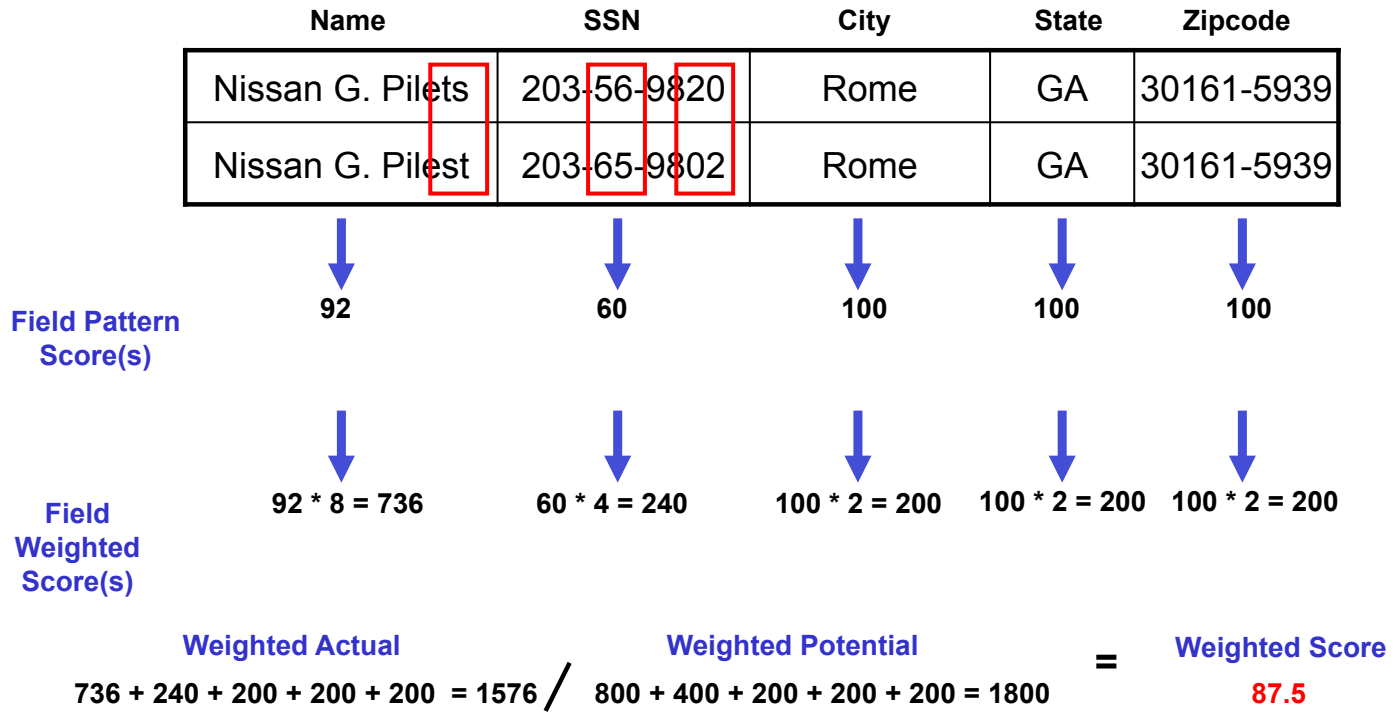


Figure 12. Match score simulation with Nissan Pilets and Nissan Pilest to demonstrate when it may not be a match.

The first thing we might consider is throwing out that field if the pattern score is less than a threshold we set. In this case we could say any score less than 80, let's not use that field in our calculations. That would yield a match score of 95.4. This makes sense.

If you do not consider the Social Security Number then these two people are very similar. You would use this rule whenever you have data that is prone to errors. Good examples of these types of fields are phone numbers and email addresses. They can help a match but if they are different it should not penalize a match.

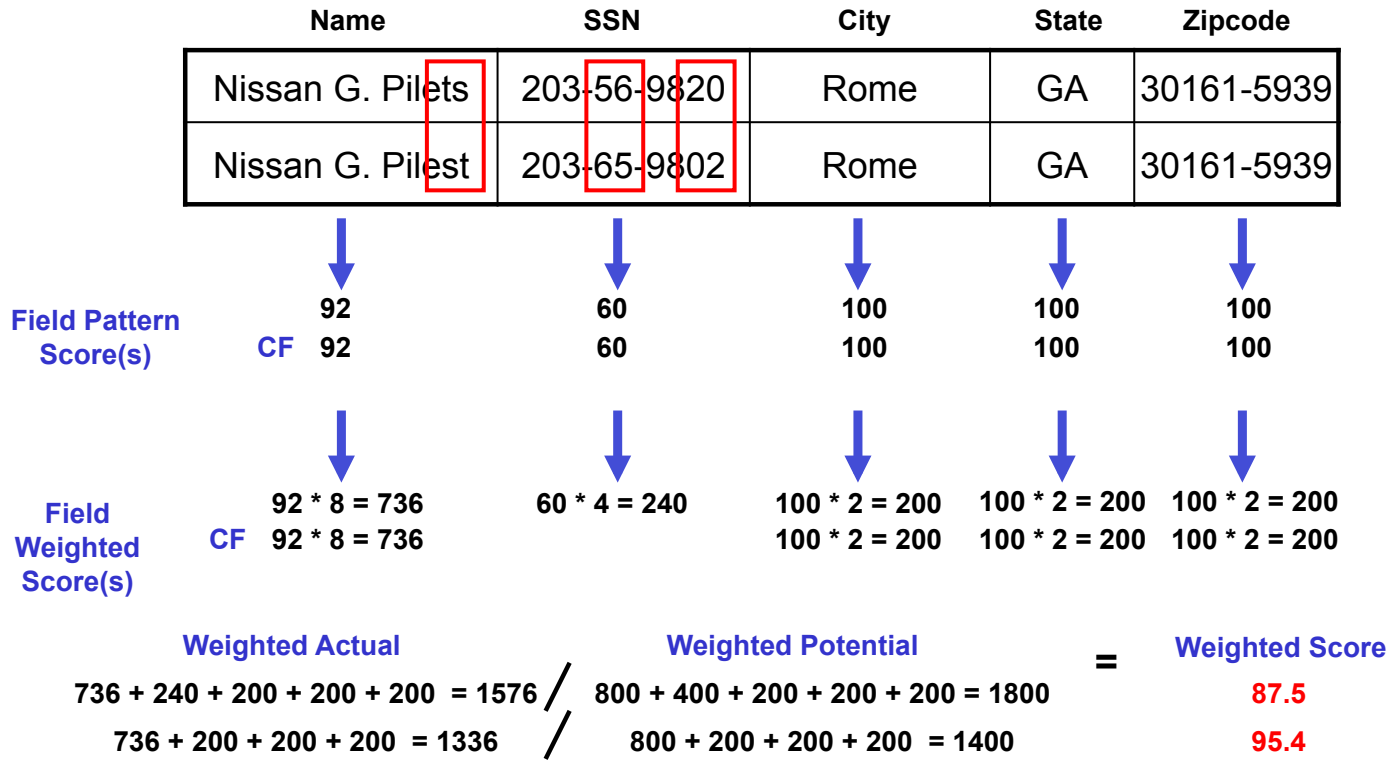


Figure 13. Match score simulation with Nissan Pilets and Nissan Pilest to show when you should not penalize a match as you would if you used a Considered Field (CF).

However, in this case, we want the fact that they have different SSN's to affect our score as we don't usually expect SSN's to be significantly different if they are the same person. But if the SSN's are different we want that difference to really impact our match score. So we set an Accentuation threshold and have any score below that threshold dropped to zero. Accentuation means that when a field pattern score is below a threshold the field weighted score is dropped to 0. (Note: Scores can also be accentuated up to 100%).

In our case rather than 240 being added to our actual score, zero is added. We also keep the potential score of 400. This will significantly affect the final match score and in our case it drops from 87.5 to 74.2 which probably more properly reflects the likelihood that these two individuals are in fact the same person.

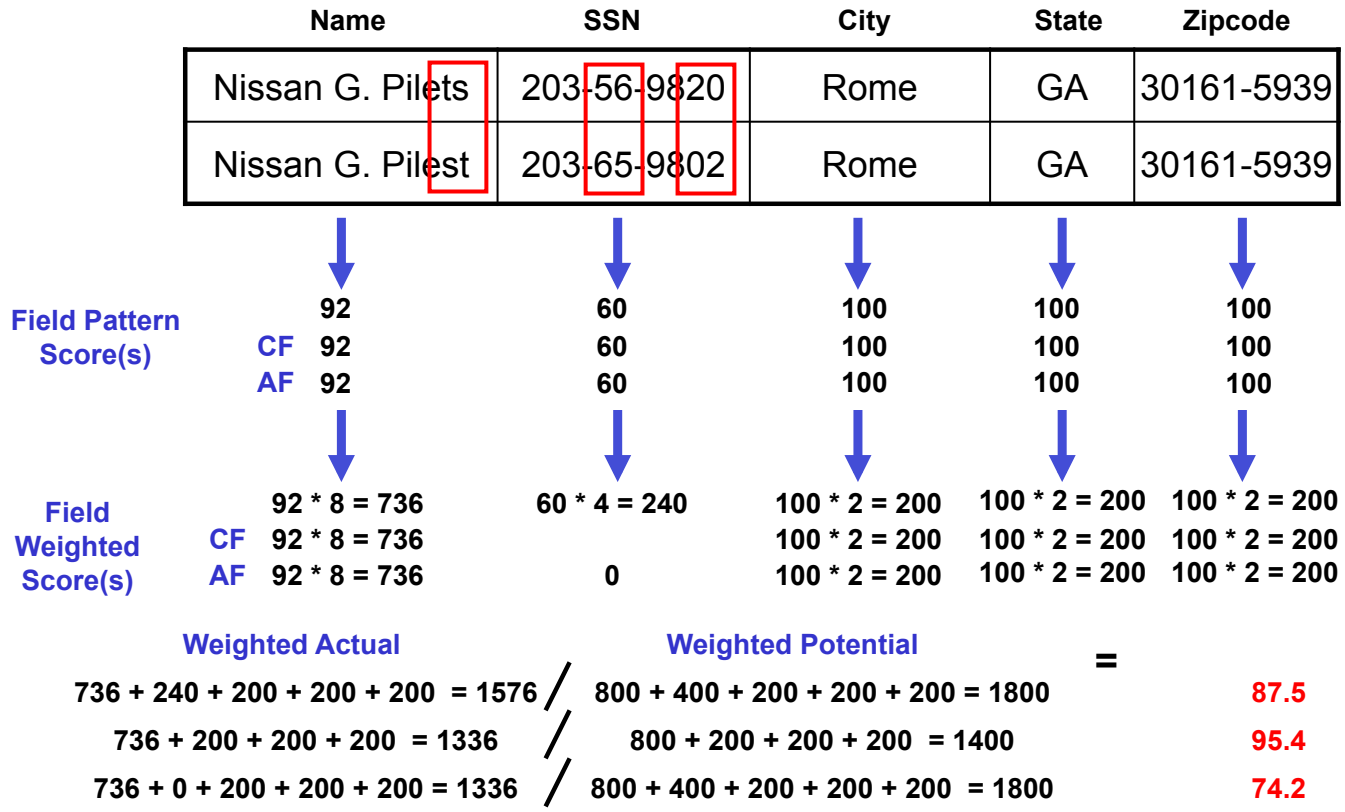


Figure 14. Match score simulation with Nissan Pilets and Nissan Pilest. A more reasonable score using an Attenuation Field (AF).

Multiple Word Match

Lastly, we must consider those fields that have multiple words within them. We have placed the first, middle, and last names in a single field. We did this so that regardless of word order the names would be properly matched. However, it is possible as in the case shown here, for one or more words to be missing.

How we calculate those missing words will affect our final match scores. If we use the second record for our reference record, the one with Nissan Pilets in the name field, then the pattern match score would be 100. This is because all of the second record is matched by the first record. This generates a final match score of 96.5.

Note for this example we corrected the last name and set our SSN back to a single transposition to make it easier to explain.

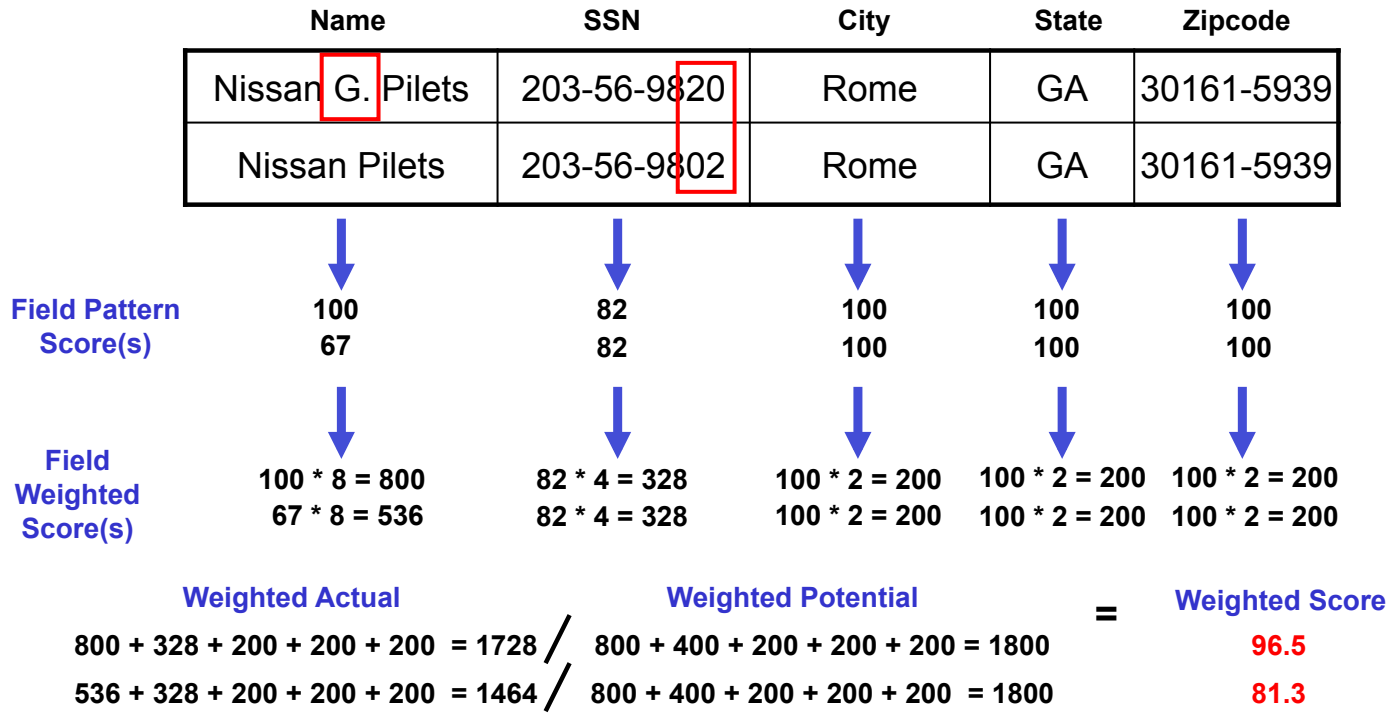


Figure 15. Match score simulation with Nissan Pilets and Nissan Pilest to show the importance of choosing a proper reference record.

However, if we use the first record, the one with Nissan G. Pilets, as the reference record then the pattern match score is 67. This is because the second record only matches 2/3 of the first record. Only the number of words is considered not the word length. This generates a final match score of 81.3.

There are four ways to designate which record to use as the reference record. The record with the most words, the record with the least words, the search record, the file record. The search record is the incoming record and the file record is your Best Version candidate. It does not really matter which one you pick as your results will be consistent and you can set your thresholds appropriately. (Note that multiple word match is more important for matching businesses as they often have multiple words in their names.)

Algorithms Used

We've discussed the calculation of the match score, a calculation consisting of the Field Pattern Scores and the Field Weighted Scores. Let's take a look at the algorithms that are behind the Field Pattern Scores.

Oracle Matching Server uses four basic algorithms: Hamming, Jaro-Winkler, Edit Distance and Bigram; each is specific to a certain type of field.

"Oracle Matching Server uses four basic algorithms: Hamming, Jaro-Winkler, Edit Distance and Bigram; each is specific to a certain type of field.."

The Hamming (a mathematician from Bell Labs) algorithm is good for telephone numbers, dates and postal codes. You need the same length fields or it doesn't perform well so parsing and standardization of those number based fields is critical.

The Jaro-Winkler algorithm is well suited for matching strings where the prefix of the string is important – company names like, ZYX Associates versus ABC Associates. This algorithm is well suited for company names.

The Edit Distance algorithm calculates the minimum number of operations needed to transform one string into another. Operations include: insertion, deletion or substitution of a character. This is optimal for short text fields and address fields.

Bigram algorithm's match score is derived from the occurrence of consecutive characters that are common to both strings. This is good for long text strings, like names.

All of these algorithms are brought to bear on the appropriate fields within the Oracle Matching Server - this is what we mean by a hybrid approach. The best algorithm is deployed against specific fields for generating a match score.

Dénouement

The process begins with loading a customer file. The records are cleansed and addresses are verified. The information then goes to the matching server where various indices are created and search nets are woven. Now when a search record is introduced, a match can be quickly performed - for the name field, a concatenation of first, middle and last name, a Bigram algorithm is applied, for the address field, an Edit Distance algorithm and for numbers such as a Social Security Number, a Hamming algorithm is used.

These various algorithms produce a Field Match score which is then multiplied with weighted field values and a Total Match Score is calculated for each candidate record. If the score is above a certain threshold, the input search record exists and can be merged to create a Best Version.

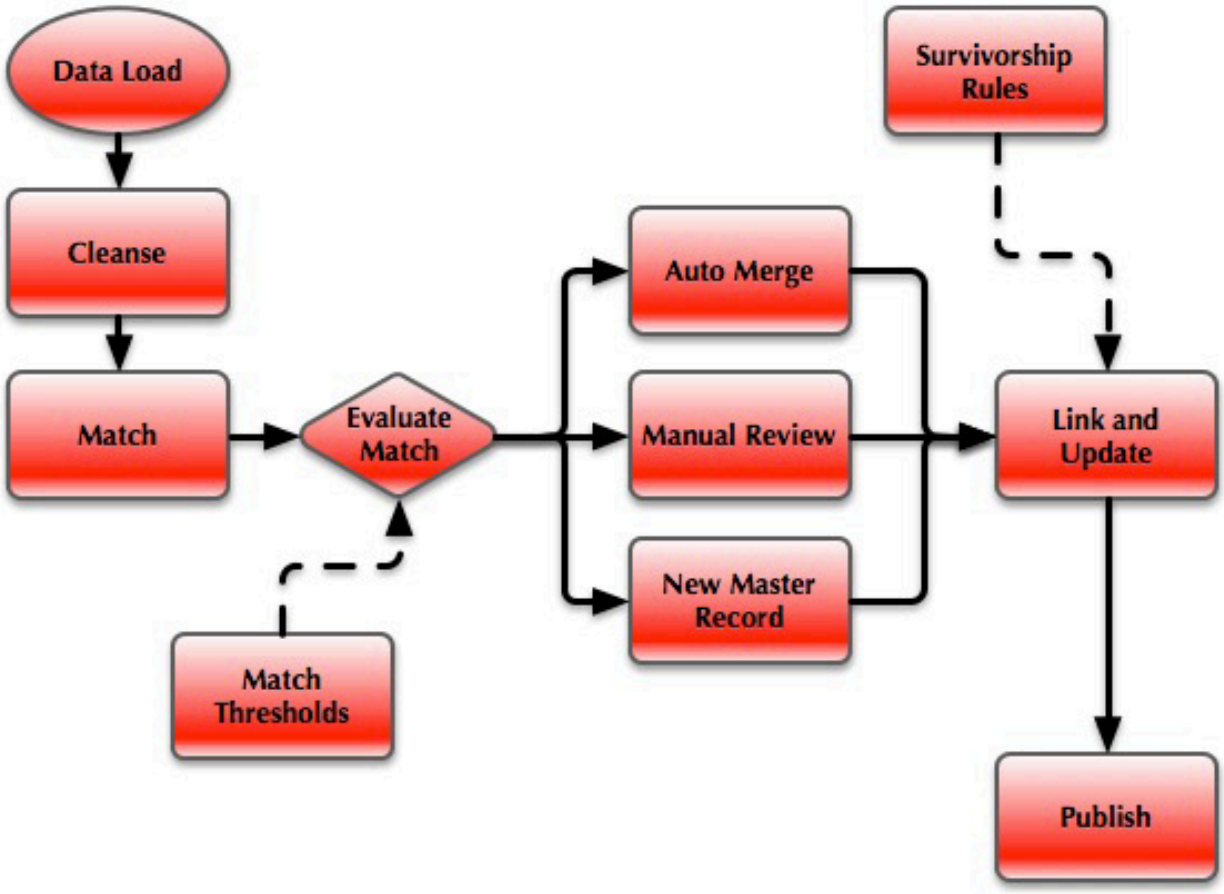


Figure 16. The Person Data Management process from beginning to end (this is a duplicate of Figure 2).

If the score is below a certain threshold, the input record does not exist and a new Best Version is created from the input record. If the Total Match Score falls between your two thresholds then the candidate record and the input record are presented for manual review.

Managing identities at the enterprise level involves a complex set of tools. A repository to store your best versions and share the information critical to your processes; a cleansing and validation engine and a search and matching engine.

With these tools you are on your way to solving the identity resolution problem and answering the question, "Who is this?" We hope this paper gives you a better understanding of the process of resolving identities and the power and flexibility of our solution to adapt to your needs.

Identity Resolution Explained
Authors: William Indest and Bruce Hamman
Superb Reviewers:
Todd Lesser and Agustin Garay
Managers:
Agustin Garay and Masood Khan
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com