Paper 3920-2015

Entity Resolution and Master Data Life Cycle Management in the Era of Big Data

John R. Talburt, University of Arkansas at Little Rock & Black Oak Analytics, Inc.

ABSTRACT

Proper management of master data is a critical component of any enterprise information system. However, effective master data management (MDM) requires that both IT and Business understand the life cycle of master data and the fundamental principles of entity resolution (ER). This paper provides a high-level overview of current practices in data matching, record linking, and entity information life cycle management that are foundational to building an effective strategy to improve data integration and MDM. These include the need for ongoing ER analytics that systematically and quantitatively measure ER performance, investing in clerical review and asserted resolution for continuous improvement, and addressing the large-scale ER challenge through distributed processing.

INTRODUCTION

The base technology that drives MDM is ER, which is sometimes called record linking, data matching, or de-duplication. ER is the process of determining when two information system references to a real-world entity are referring to the same, or to different, entities (Talburt, 2011). ER represents the "sorting out" process when there are multiple sources of information that are referring to the same set of entities. For example, the same patient may be admitted to a hospital at different times or through different departments such as in-patient and out-patient admissions. ER is the process of comparing the admission information for each encounter and deciding which admission records are for the same patient and which ones are for different patients.

If two references refer to the same entity, they are said to be "equivalent" references. When an ER system makes the decision that two references are equivalent, it represents that decision by adding an extra value to each reference called a "link". The actual value of the link is not important, only that references judged to be equivalent have the same link value, and reference thought to be non-equivalent have different link values. For this reason, ER is sometimes called "record linking." The fundamental law of ER states that two references should be linked if, and only if, they are equivalent. However, ER systems always make some mistakes. They sometimes link two references that are not equivalent (a false positive error), and fail to link two references that are equivalent (a false negative error).

Although ER is necessary for effective MDM, it is not, in itself, sufficient to manage the life cycle of identity information. Entity Identity Information Management (EIIM) is the collection and management of identity information with the goal of sustaining entity identity integrity over time (Zhou & Talburt, 2011c). Entity identity integrity requires that each entity must be represented in the system one, and only one, time, and distinct entities must have distinct representations in the system (Maydanchik, 2007). Entity identity integrity is a fundamental requirement for MDM systems.

EIIM is an extension of ER in two dimensions, knowledge management and time. The knowledge management aspect of EIIM relates to the need to create, store, and maintain identity information. The knowledge structure created to represent a master data object is called an entity identity structure (EIS).

The time aspect of EIIM is to assure that an entity under management in the MDM system is consistently labeled with the same, unique identifier from process-to-process. This is only possible through EIS that store the identity information of the entity along with its identifier so both are available to future processes.

ASSESSING ER RESULTS

If the goal of EIIM is to attain and maintain entity identity integrity, then it is important to be able to measure attainment of the goal caused by false positive and false negative errors made by the ER process. Unfortunately, in many organizations the assessment of ER and EIIM results is done poorly if at all. ER and MDM stewards often rely entirely on inspection to see if the matching seems reasonable.

However, inspecting references for quality of match is inadequate in two ways. First, not all matching references are equivalent, and second, equivalent references do not always match.

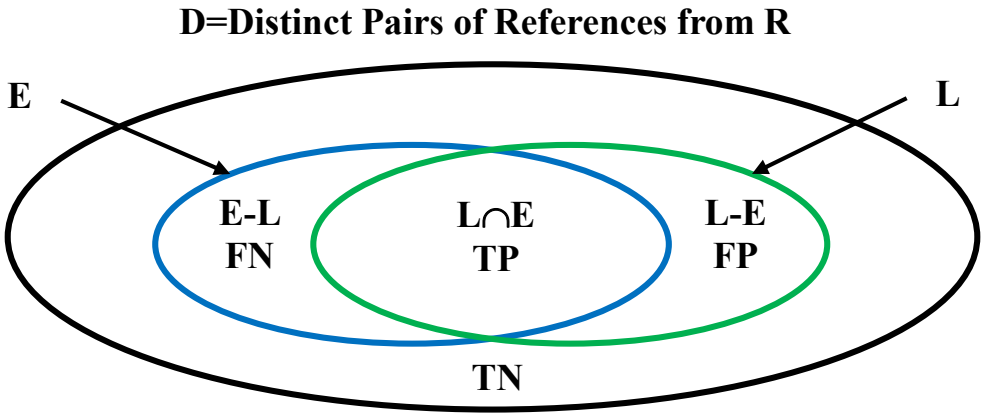
Just inspecting clusters of matching references does not actually verify that the references are equivalent. Verification of equivalence is not the role of a matching expert, it should be done by a domain expert who is in a position to know if the references really are for the same real-world object. Furthermore, cluster inspection leaves out entirely the consideration any equivalent references that should be in the cluster, but were missed by the ER process.

In order have a realistic assessment of ER results, the visual inspection of match clusters should be replaced by, or at least supplemented with, other more comprehensive and systematic methods. Two commonly used approaches to this issue often used together are truth set development and benchmarking (Syed, et al, 2013).

TRUTH SETS

A truth set is a sample of the reference for which entity identity integrity is known to be 100%. For a large set of references R, a truth set T is a representative sample of reference in R for which the true equivalence or non-equivalence of each pair of references has been verified. This verification allows the references in to be correctly linked with a "True Link." The true link identifier must be assigned and verified manually to make sure each cluster of references with the same true link value reference the same entity and that clusters with different true link values reference different entities. In other words, the truth set obeys the fundamental law of ER.

Assessment using the truth set is simply a matter of performing ER on the truth set, then comparing the clusters created by the ER matching process to the True Link clusters. If the ER link clusters coincide exactly with the True Link clusters, then the ER process has attained entity identity integrity for the references in the truth set. If a cluster created by the ER process overlaps with two or more truth set clusters, then the ER process has produced some level of linking errors.



References (R)	Linked Pairs (L)	Not-Linked
Equivalent Pairs E	**True Positive** **L ∩ E**	**False Negative** **E – L**
Not Equivalent D – E	**False Positive** **L – E**	**True Negative** **D – (L ∪ E)**

Figure 1. Venn Diagrams of the ER Linking Outcomes for a Set of Reference R

The Venn diagrams shown Figure 1 illustrates the relationship among the linking outcomes when an ER process is applied to set of reference R. The universe D is the set of all un-ordered pairs of references in R. If R has N records, then D will contains N(N-1)/2 pairs. The subset of D labeled L represents all of the pairs in D that were linked together by the ER process, and the subset E represents all of the pairs in D that are actually equivalent.

The intersection of L and E are called “true positives” because they are equivalent and thus correctly linked. All of the pair outside of the union of L and E (the figure eight shape) are called “true negatives” because they are not equivalent (outside of E) and were not linked (outside of L). The errors are the false positives, the pairs in L that are not in E, and the false negatives, the pairs in E that are not in L. The ER process is 100% accurate if both FP and FN are empty. In general, the linking accuracy of the ER process is given by

$$Linking\ Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$

There are also several other common measure of ER outcome that can be computed from these same parameters (Christen, ?). These include

$$False\ Negative\ Rate = \frac{|FN|}{|TP| + |FN|}$$

$$False\ Positive\ Rate = \frac{|FP|}{|TN| + |FP|}$$

$$Linking\ Precision = \frac{|TP|}{|TP| + |FP|}$$

$$Linking\ Recall = \frac{|TP|}{|TP| + |FN|}$$

BENCHMARKING

Benchmarking is another type of ER outcome assessment. It is the process of comparing one ER outcome to another ER outcome where the ER processes are acting on the same set of references. Even though benchmarking is a relative measure, it is useful because most organizations trying to develop a new MDM system rarely start from scratch. Usually, there is a legacy system. Benchmarking is similar to the truth set approach because some previous ER result is a stand-in for the truth set. However, when a new ER result is compared to the benchmark result and overlaps are found, it is not which is more accurate, only that they are different.

Benchmarking assessment is based on the following assumption: when both systems agree, those links are the most likely to be correct. This assumption allows the evaluation process to focus on references linked differently in the benchmark than in the new process. These differences must still be verified, but this method considerably reduces the review and verification burden. Where there are a large number of differences, a random sample of the differences of manageable size can be taken to estimate whether the differences indicate that the new process is making better or worse linking decisions than the benchmark.

Benchmarking and Truth Set evaluation are often used together because neither is an ideal solution. Because truth sets tend to be relatively small, they may not include relatively infrequent data anomalies or reference configurations particularly problematic for the ER process. The obvious advantage of the benchmark is it can include all of the references in production at any given time. The problem with benchmarking is its lack of precision. Even though links made by both systems are assumed to be correct, they may not actually be correct.

TALBURT-WANG INDEX

The Talburt-Wang Index or TWi is another measurement that combines both FP and FN errors into one number (Talburt, Wang, et al, 2007). It is a useful metric for benchmarking because its computation is much easier for large datasets, and it does not require the calculation of the individual TP, TN, FP, PN counts. It takes advantage of the fact that when an ER process links a set of references R, it creates a partition of R. A partition is simply a set of non-empty, non-overlapping subsets whose union is the entire set. For a set of references R, the subsets where each subset comprised the references that have the same link value, will form the partition of R. The TWi is based on the number of overlaps between two partitions formed by different ER linking processes.

If R is a set of references, and A and B represent two partitions of R created by separate ER processes, then define V, the set of overlaps between A and B as

$$V = \{A_i \cap B_j | A_i \cap B_j \neq \emptyset\}$$

Then the Talburt-Wang Index is defined as

$$TWi(A,B) = \frac{\sqrt{|A| \cdot |B|}}{|V|}$$

Two important characteristics of the TWi are

$$0 < TWi(A,B) \leq 1.0$$

$$TWi(A,B) = 1.0 \; if \; and \; only \; if \; A = B$$

In the case A represents the partition formed by true links (verified correct links), then the TWi(A,B) is an alternative measure of linking accuracy of B.

THE ROLE OF CLERICAL REVIEW

The concept of clerical review has been a part of entity resolution and record linking since its beginning with the Fellegi-Sunter Theory of Record Linking (Fellegi, Sunter, 1969). An essential element of their proof was that in order for a match rule to meet a given constraint on the maximum allowable false positive and false negative rates, some minimal number of reference pairs would have to be manually reviewed by a domain expert who could correctly classify them as being equivalent or not equivalent. In the case of the Fellegi-Sunter model, the review indication occurs when a pair of references satisfied a particular agreement pattern.

For example, in matching student enrollment records, the pattern of disagreement on first name, but agreement of last name, date-of-birth, and house number might fall into this review category. The pattern strongly suggests the records may be referencing the same student but also the possibility the records satisfying this match pattern are twin siblings. Sometimes these are called soft rules or weak rules. Instead of linking the references, the firing of these rules signals that the pair of records should undergo clerical review. In probabilistic matching based on scoring rules, review indication is triggered when the match score falls below the match threshold, but above the review threshold.

In EIIM, the ER linking results are saved in as a data structure called an “entity identity structure” or EIS. The EIS are created by an automated process that tests each pair of references may some type of matching rule. But no matter how carefully these rules are crafted, they will always produce some level of false positive and false negative error. Clerical review is a manual process that adjusts the EIS when these errors are discovered. For a system of any size, the reviewer will need two things; a tool to assist in viewing the EIS in order make a review decision, and a mechanism for making an adjustment (update) to the EIS if an error is present.

ASSERTED RESOLUTION

Asserted resolution, sometimes called informed linking or knowledge-based linking, is simply the re-configuration the EIS based on expert knowledge. Assertions are effected by assertion transactions applied to the EIS by the EIIM system. Because asserted resolution is driven by a manual review

process, the number of assertions applied will always be small when compared to the large volume of updates applied through the automated update process. Clerical review and assertion are important to reduce the accumulations of false positive and false negative errors. Left unattended, these errors can build up over time in an un-reviewed EIIM system. Assertion complements the automated ER process and provides a mechanism for continuous improvement of the EIIM system.

Direct manipulation of the IKB through an editing tool or other ad hoc processes is never a good idea and should be prohibited by MDM governance policy. All update transactions to the IKB including assertion transactions actions should always be mediated through thoroughly tested application software logging transaction events to facilitate process auditability and to maintain entity identity integrity.

Correction Assertions

There are two categories of assertions – correction assertions and confirmation assertions. Correction assertions are used to correct FP and FN errors in and between EIS. Confirmation assertions do not alter EIS other than to insert metadata flagging the EIS as being correct.

Structure-to-Structure Assertion

A structure-to-structure assertion is used to correct a false negative error in which two EIS are found to be equivalent, i.e. reference the same entity. Although some false negatives are self-correcting when new references connect the EIS during the identity update process, this is not always the case, and manual intervention may be required. Figure 2 shows the schematic of a structure-to-structure assertion.

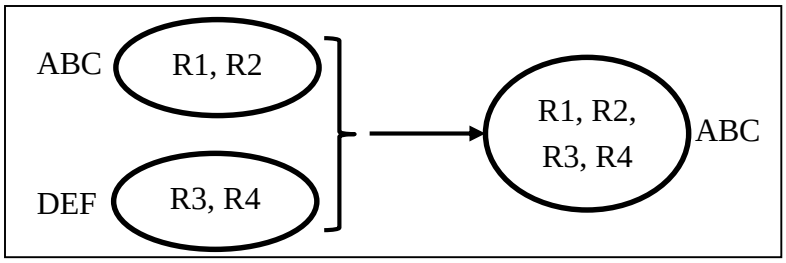


Figure 2: Structure-to-Structure Assertion

In Figure 1, ABC and DEF are forced to merge into a single cluster. To minimize entity identifier changes, the identifier for one EIS is retained, in this case ABC. Only entity identifier DEF will have to be retired. Although Figure 1 only shows two EIS being merged, some systems support merging several equivalent EIS in the same assertion.

In the context of MDM, the retirement of identifier DEF will orphan any source records residing in client systems still referencing DEF. These records should now reference ABC. The mechanism making this change in reference (synchronization) will depend upon the MDM architecture of the system. For example in a Registry Architecture, the client systems will have to push their records to the MDM hub in order to refresh its identifiers, whereas in a Reconciliation Engine, the hub will notify client systems holding references to DEF that they should no reference ABC.

Structure-Split Assertion

The structure-split assertion is designed to correct false positive errors, i.e. EIS containing references to more than one entity. Unlike false negative errors, false positive errors are never self-correcting through the automated update process. ER processes are driven by matching engines only making decisions to merge EIS, never to split them. Once a false positive EIS is created, it will remain in the system until there is a manual intervention to correct it.

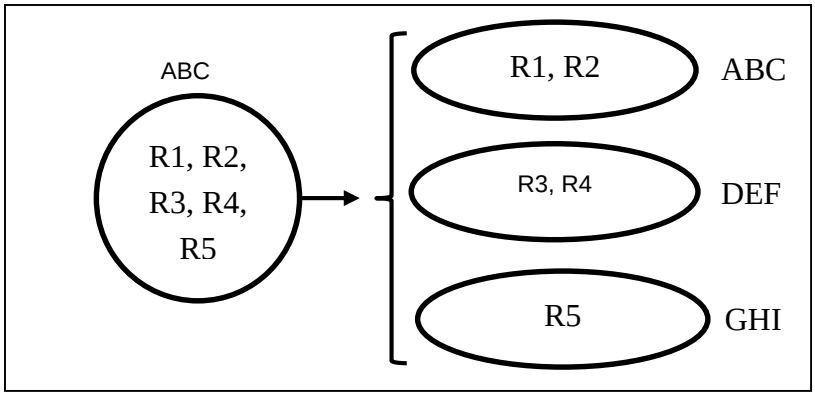


Figure 3: Structure-Split Assertion

A schematic for the structure-split assertion is shown in Figure 3. In this example the EIS ABC contains five references R1, R2, R3, R4, and R5. A clerical review shows references R1 and R2 reference one entity, references R3 and R4 reference a second entity, and reference R5 references yet a third entity. Correcting this problem requires a set of structure-split assertion transactions. Again, to minimize the amount of effort needed to synchronize the client systems, the MDM system will not allow all of the references to be split away from the original EIS. In other words, the system requires the original EIS must survive the structure-split assertion operation, so the original entity identifier ABC is retained in the system. The structure-split assertion transactions only need to call out the references needing to be moved out of EIS.

The synchronization of identifiers in an MDM client system for structure-split assertions is more difficult than for the structure-to-structure previously discussed. The problem now is clients' systems may contain many source records referencing identifier ABC, but after the assertion, some of these perhaps should reference the new identifier GHI. Even if the system has kept track of which source records were appended with identifier ABC, it is not obvious which of these should be changed to GHI without having to re-match the records against the identity information. This is another reason why ER systems are usually tuned to prefer false negative errors over false positive errors. Not only do the corrections to false negative errors stay corrected without special logic, the identifier retirement is simpler to manage.

Confirmation Assertions

Confirmation assertions are designed to label EIS as having been reviewed and confirmed as correct in order to prevent their continued review. The algorithms producing clerical review exceptions are not perfect. Some EIS called out for clerical review as potential false positives often turn out to be true positives, i.e. correctly clustered. Similarly, some groups of EIS called out for clerical review as false negatives are in fact true negatives. Without confirmation assertions, these EIS can be repeatedly called out for review by the clerical review indicator logic even though they have previously been manually reviewed and found to be correct.

However, it is important to note EIS reviewed as correct should only be excluded from subsequent reviews as long as they maintain the state they had at the time of the review. Any changes to the EIS could change their status from correct to incorrect. The implementation of confirmation assertions also requires new functionality in the ER update logic that will remove confirmation labels from the EIS metadata whenever the ER process modifies the EIS.

True Positive Assertion

The true positive confirmation assertion pertains to a single EIS. If an EIS is called out for clerical review, and if the reviewer finds all references in the EIS are for same entity, the EIS is a true positive and should be asserted as such. The true positive assertion of an EIS will add a metadata label showing the EIS is true positive.

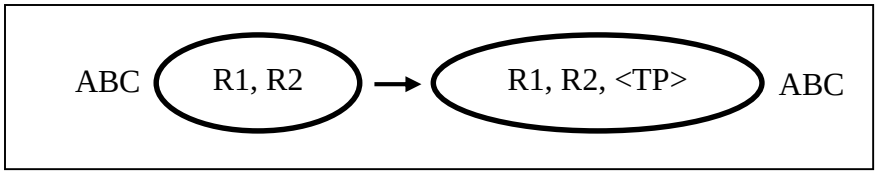


Figure 4: True Positive Assertion

The case shown in Figure 4 is where the EIS with entity identifier ABC was called out for clerical review as a false positive, but after inspection it was found to be correct. The action of the true positive assertion is to add a metadata tag (shown as <TP> in Figure 4) to the EIS. The true positive tag inserted into ABC will prevent the clerical review indicator logic from calling out ABC for review as long as the tag is present.

However, if a later update process results in new references being added to ABC, then the true positive tag will be removed, and ABC may again be called out for clerical review in later processing. The metadata added to the true positive EIS may also include other information for auditing and traceability such as an identifier for the person making the review decision, the date of review, and other relevant information. These additional metadata are important for good master data governance.

True Negative Assertion

The True Negative Assertion confirms two or more EIS were called out for review as potential false negatives have been confirmed as correct by a reviewer. Just as with the true positive assertion, the true negative assertion inserts metadata tags into the reviewed EIS. In the case of true negatives, additional metadata is required because a true negative state always exists between two (or more) EIS. Just labeling an EIS as a true negative does not make sense by itself. A true negative assertion of an EIS must be expressed in relation to another EIS. In addition to a true negative label, a true negative assertion must also insert the identifier of the EIS to which it is a true negative. These metadata must be inserted into all of the EIS the review process determines to be true negatives of each other.

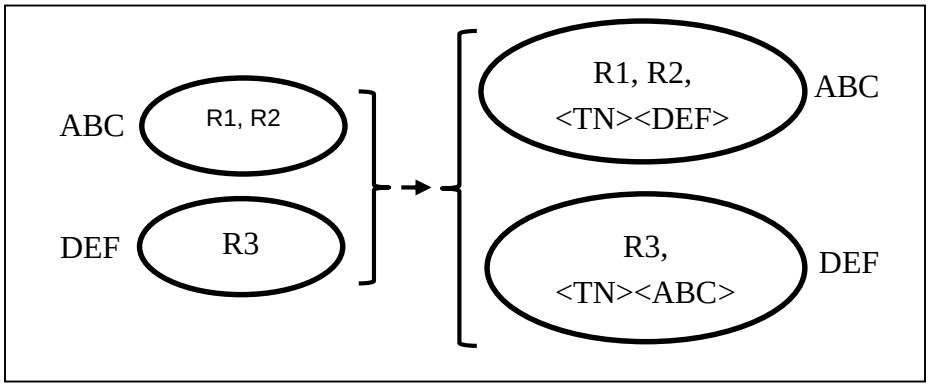


Figure 5: True Negative Assertion

As shown in Figure 5, the true negative assertion of the EIS with identifiers ABC and DEF creates metadata that cross-references these EIS. The EIS with identifier ABC cross-references the EIS with identifier DEF as a true negative, and conversely, the EIS with identifier DEF cross-references the EIS with identifier ABC as a true negative. These tags will prevent the clerical review logic from calling out the EIS identified as ABC and DEF as false negatives in future processes as long as they maintain the original state. Just as with the true positive assertion, the metadata tags suppressing subsequent true negative review must be removed if and when a later update process adds new references to any one of the EIS in a true negative group.

DISTRIBUTED PROCESSING FOR LARGE-SCALE ER

ER is fundamentally an $O(n^2)$ problem. All ER systems must use some type of “blocking” strategy to reduce the number of comparisons required to resolve the input into equivalent references. Blocking simply means performing the ER process on specific subsets of the input instead of on the entire set

references. These subsets are called blocks. For example, if the input has 1,000 references. If an ER process is performed on the entire 1,000 references, then the processing will require approximately $1{,}000^2$ or 1,000,000 comparisons in total. But if the references are partitioned into 10 non-overlapping blocks of 100 references each, then the ER algorithm will make approximately 100^2 or 10,000 comparison for each block. Over all 10 blocks the total effort will be approximately 10 x 100^2 or 100,000 comparisons, an entire order of magnitude less that performing the ER algorithm on the entire 1,000 references.

Several approaches to blocking have been developed. Christen (2012), Isele et al, (2011) and Baxter et al (2003) describe a number of these including inverted indexing, sorted neighborhoods, Q-gram indexing, suffix-array indexing, canopy clustering, and mapping-based indexing. However, inverted indexing, sometimes called match key indexing or standard blocking (Christen, 2006), is one of the most common approaches to blocking in ER systems.

A simple example of match key blocking for customer references with name and address fields would be to block the references by zip code. The value used to partition references into blocks, such as the zip code, is called the match key. Sometimes as in this example, the match key is simply the value of a single attribute. In other cases, the value may be modified in some way or concatenated with other values. The routines that create match keys are called match key generators.

Match key blocking is one of the primary strategies for addressing the problem of large-scale ER using Hadoop Map/Reduce and related technologies (Bianco et al, 2011; Kirsten et al, 2010; Yancy, 2007). The Hadoop Map/Reduce is a processing layer that sits on top of the Hadoop File System (HDFS). In the Hadoop architecture, a mapper function outputs key-value pairs. The system then brings together all of the key-value pairs into blocks that share the same key. These blocks are then processed by reducer functions. This paradigm fits nicely with match key blocking in which the key of the key-value pair is the match key, and the value of the key-value pair is the reference.

Large-Scale ER with Single Match Key Blocking

Suppose for a base ER rule there is a single, properly aligned index generator. Alignment of the match key generator with the match rule implies all pairs of references that match by one of the rules also generate the same match key value. Therefore, from an entity resolution standpoint, this means all of the matches that can be found by the match rule are going to be found between references within the same block. By the assumption of alignment, references from different match key blocks should not match. Consequently in the case of single index generator, the complete resolution of a dataset can be accomplished in relatively simple, two-step process as shown in Figure 6

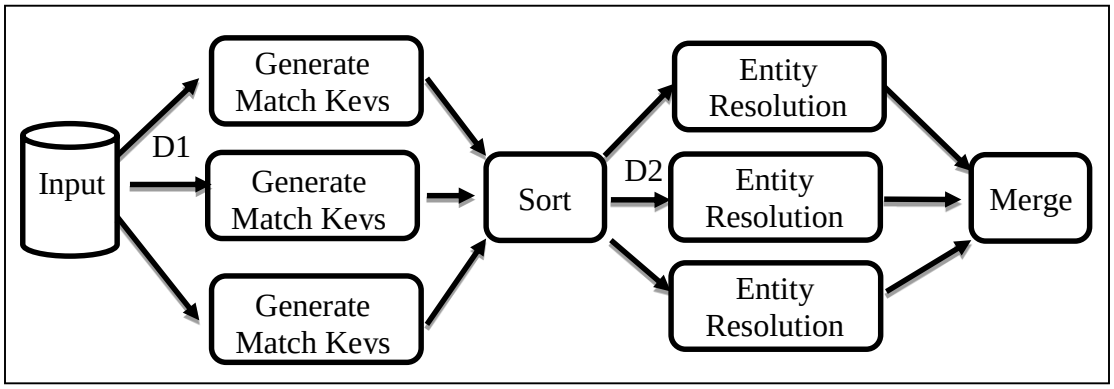


Figure 6: Distributed Workflow based on Single Index Generator

The first step is the generation of the match key values. At this first step, the partitioning and distribution D1 of the input references to different processors is arbitrary and can simply be based on uniformly sized groupings of references. This is because the match key value produced by a match key generator only depends upon the values of the identity attributes in each reference, and there are no dependencies on other references. In the Hadoop Map/Reduce framework, algorithms such as match key generation that operate independently on individual records are usually run in a map step. The output of a map step is

always a key-value pair. In this case the key of the key-value pair is the generated match key, and the value of the key-value pair is the reference that generated the match key.

After each block of input references has been processed by the match key generator running at each of the nodes, each reference will produce a single match key. These match key-reference pairs are then merged and sorted into match key value order. In the Hadoop Map/Reduce framework sorting by key value is performed by a shuffle step that follows the map step and precedes the reduce step. Unless otherwise specified, the shuffle sorts the mapper output in key value order by default.

After the references are sorted into match key order, Hadoop automatically creates key-blocks. Each key-block comprises all of the key-value pairs that share the same key value. In this case, it will be blocks of references where all of the references in the same block share the same match key value.

In the reduce step, the key-blocks are distributed to different processors where the final resolution of blocks into clusters takes place. Again, this fits well with the Hadoop map/reduce framework that guarantees all of the records in a key-block are processed together on the same processor. One processor could be given multiple blocks, but the reducer function of Hadoop assures blocks are not split across processors.

Once a block of key-value pairs is sent to a processor, it can be processed by an ER algorithm and base match rule to link the references into clusters. Consequently, in a Hadoop framework the ER process itself works best in a reduce step. Taking again the example of blocking by postal code, the key of the key-value pair would be the postal code and the value of the key-value pair would be the actual reference. In this way, all references with the same postal code would be sent to the same processor. The reduce step guarantees blocks with the same postal code will be sent to the same processor provided the postal code block does not exceed the capacity of a single processor. Typically match key blocks will be smaller than the single process capacity allowing the system to send several key blocks to the same processor. The reducer provides an iterator function that allows the code resident on the processor to access each key-value pair in the key-block sequentially. In the final reduce step each match key block becomes the input to an ER algorithm that will further refine the block into clusters according to the matching rules of the ER process.

The Transitive Closure Problem

However, there is a problem associated with the process described in the Figure 6. The problem is that in most large-scale entity resolution applications, a single match key is not sufficient to bring together all of the references that should be compared. Data quality problems such as typographical errors in attribute values, missing values, and inconsistent representation of the values will require the use of more than one match key in order to obtain good resolution results with a reasonable amount of comparison reduction.

When each input reference can potentially create more than one match key, the processing method outlined in Figure 6 will not always lead to the correct resolution of the references in the input. The reason is the requirement for transitive closure of an ER process. Transitive closure states if a reference R1 is linked to a reference R2, and reference R2 is linked to a reference R3, then reference R1 should be linked to reference R3. If the ER system does not obey this transitive closure rule, it can give inconsistent results, i.e. the outcome of the ER process can be dependent upon the order in which the references are processed.

The workflow for multiple match key resolution as shown in Figure 7 begins in the same way as the process outlined in Figure 6 for a single index workflow. The problem is that simply sorting and distributing by match key blocks is no longer adequate.

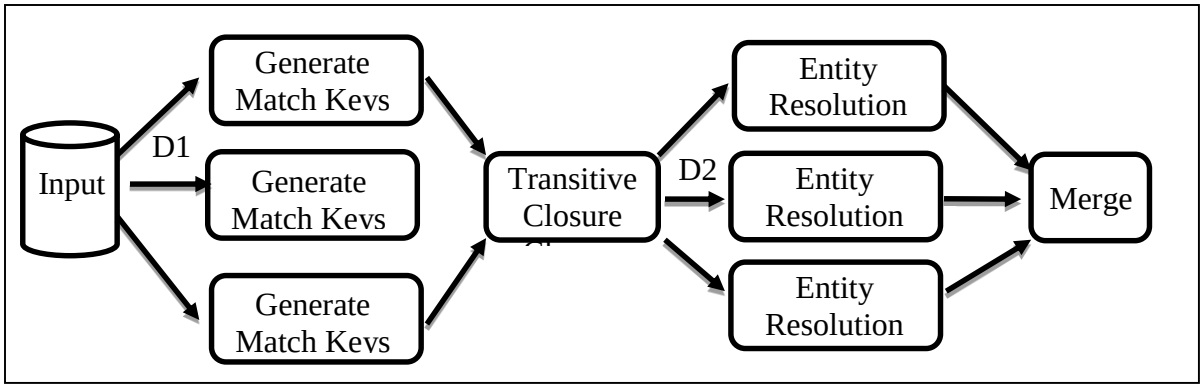


Figure 7: Distributed Workflow for Multiple Match Keys

One solution to the problem shown here is to replace the sort process with a transitive closure process where the closure is with respect to shared match key values. This can most easily be understood by looking at the references and relationships in the form of a graph.

Transitive Closure as a Graph Problem

In graph theory an undirected graph is simply a set N of nodes and a set of E of edges, where E is a subset of P(N), the set of all subsets of N that contain two elements. As a simple example consider, N = {a, b, c, d, e, f} and E = {{a, b}, {b, c}, {d, e}}. N and E form an undirected graph of six nodes and three edges depicted in Figure 10.3.

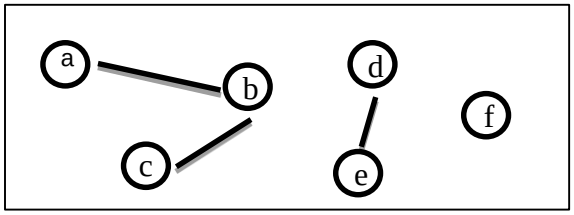


Figure 8: Undirected Graph of N and E

A connected component of a graph is a maximal subset of nodes in the graph having the property for any two nodes in the subset, there exists a path of edges that connects them together. In Figure 8, it is easy to see the graph has three connected components {a, b, c}, {d, e}, and {f}. Even though nodes “a” and node “c” do not share an edge, there is a path from “a” to “b” then from “b” to “c” that connects them. It will also be true the connected components of a graph form a partition of the nodes, i.e. every node in the graph is in one, and only one, component.

The connected components of a graph can easily be found using a recursive, breath-first search algorithm (Sedgewick, Wayne, 2011). However, there is a limiting feature of this algorithm when working with Big Data. The recursive algorithm for transitive closure requires in-memory access to every node and edge of the graph to work efficiently. This may not be practical or even possible for very large datasets. To solve this problem, several iterative, non-recursive algorithms for transitive closure have been developed to solve this problem. For example, Talburt and Zhou (2015) described one that runs in the Hadoop Map/Reduce environment.

The Large Component Problem

Although the pre-resolution transitive closure workflow shown in Figure 7 works, as a practical matter it will only work well when the match keys are similar to, or the same as, matching rules. The problem is not with logic of the system, but the way in which the software and the data interact. The problem is that the components (blocks) can become very large and overwhelm the capacity of any one single processor in the distributed processing environment. If this happens, the job will fail.

Solving the problem of large components in transitive closure requires some way of combining the transitive closure process with the matching process. Working together they can prevent the formations of

long chains in the graph that create the large components. Various methods have been proposed for controlling the growth of components (Kardes, et al, 2013) (Papadakis, et al, 2012) (Kolb, Thor, Rahm, 2011). Two of these are post-resolution transitive closure and incremental transitive closure.

Post-Resolution Transitive Closure

In post-resolution transitive closure each match key group is closed into an EIS with an entity identifier. After each match key is closed, then the transitive closure these entity identifiers creates the final EIS as shown in Figure 9

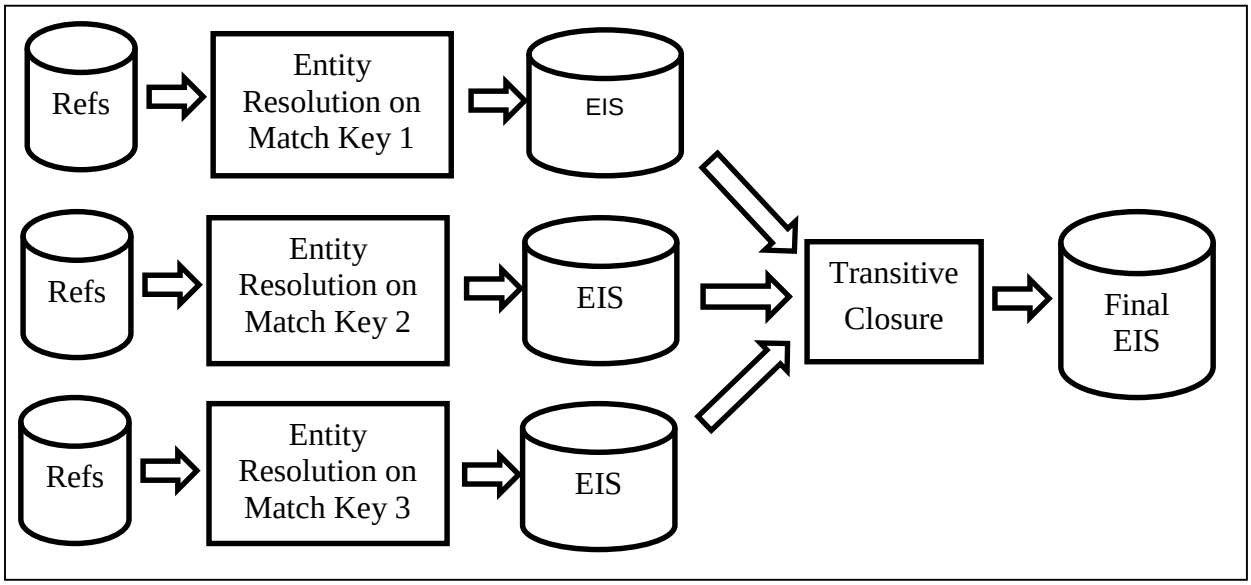


Figure 9: Post-Resolution Transitive Closure

Here the same set of references (Refs) is resolved multiple times, one time for each match key. Each individual match-key resolution corresponds to the workflow shown in Figure 6. However, the ER process uses a full set of matching rules, so the clusters formed at the end of each resolution step comprise only equivalent references. In the transitive closure process, the chaining will only be between the entity identifiers assigned to each reference rather than on the match keys. The components formed in the closure process are the EIS that would be formed using all of the match keys and matching rules in one ER step. However in this process, the components formed in the transitive closure are constrained to be no larger than the largest entity that can be formed by the matching rules.

The advantage of post-resolution transitive closure is that the maximum component size for each match key is known in advance, and the growth of the components in the final transitive closure is constrained to only chain equivalent references. The obvious disadvantage is that the full set of input references must run N times where N is the number match key generators. Also, the input to the final transitive closure is N times the size of the original set of input references.

Incremental Transitive Closure

The incremental transitive closure process is illustrated in Figure 10.

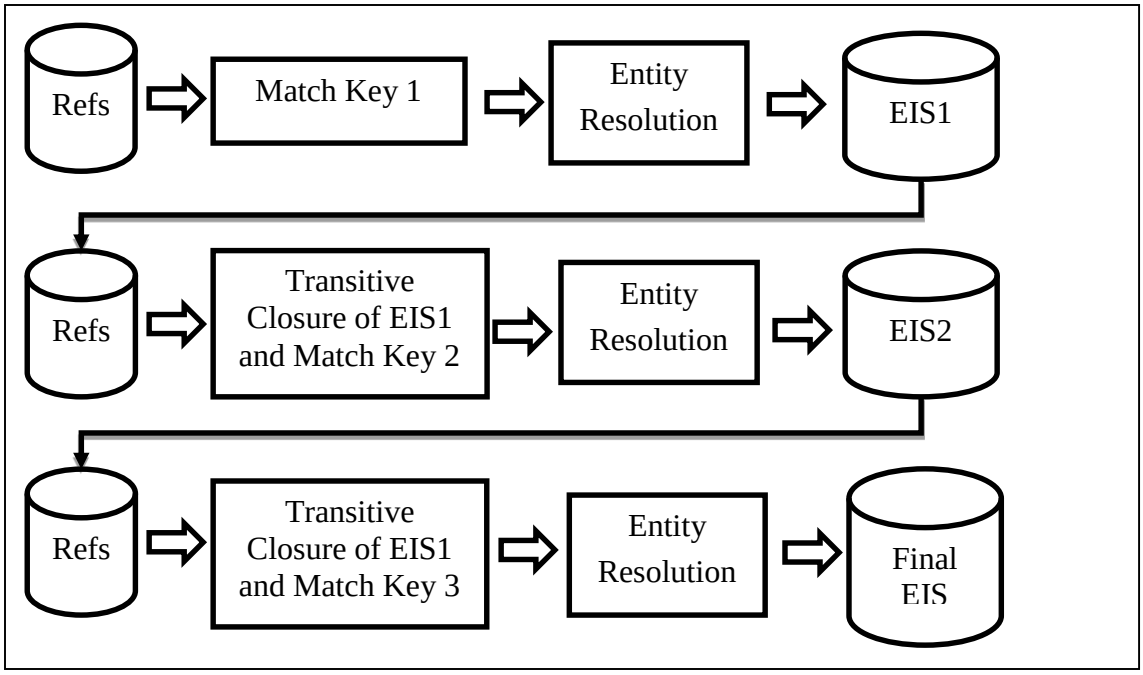


Figure 10: Incremental Transitive Closure

The incremental transitive closure process starts in the same way as post-resolution transitive closure where the ER process uses only the first match key generator. In the second step, the entity identifier assigned to each reference by the ER process is appended as an additional attribute to each of the original references. In addition, the next match key generator is applied so each reference has two match keys- the entity identifier from the previous ER process and the match key from the next match key generator. The transitive closure operates on these two keys followed by the full ER process that assigns a new set of entity identifiers. This process continues until all of the match keys have been processed.

The advantage of the incremental process is both the transitive closure process and the ER process can be highly optimized. In the transitive closure process, the bootstrap phase can take advantage of the previous matching. For example, if a group of references sharing the next match key also have the same identifier from the previous ER process, then the group does not need to go through the closure process because the match key is not adding any value because is not bringing together any references that have not already been determined as equivalent. Similarly in the ER process, any references already determined to be equivalent in a previous ER process (i.e. share the same entity identifier) do not need to be matched again. Matching only needs to take place between references that have different entity identifiers and share the same match key.

Other Challenges

In addition to the big component problem, there are several other challenges in moving traditional ER and MDM architectures into a distributed processing environment. Two of these are how to represent EIS in a distributed data store, and the problem of updating EIS with new information to complete the EIIM life cycle. Solutions for these problems are outlined here, but the full exploration of these topics is too long for this paper. Additional details about these and other topics discussed in the paper can be found in Talburt & Zhou (2015).

In traditional MDM systems, the EIS corresponding to the entities under management are stored in a relational database schemas where the each component is in a separate table and the tables are related using foreign key relationships. However, most large data stores such as HDFS, Hbase, and MongoDB are organized as folders of key-value pairs. Because these are distributed file systems, they only presents the user with a logical view of a file as a folder. Underneath the folder the system may be managing the physical file as segments distributed over several different underlying storage units.

Also in a relational database model, the logical items are parsed into separate values at the time they are written into a table. Each item value occupies a cell at the intersection of row and one column of a table. In the key-value paradigm the logical items remain blocked together in the single value of the key-value pair. The individual items must be parsed out into individual values each time the key-value pair is read from the system.

In the HDFS distributed data store, both data and metadata reside in folders created by map/reduce processes. Each folder corresponds to a logic dataset comprising key-value pairs. The physical segments of the dataset may actually reside in different locations managed by HDFS.

The Update Problem

Another important EIIM configuration is the automated update. When new references are introduced, the system uses the matching rule to decide if each new reference is referring to prior EIS, or if it is referring to some new entity not yet represented in the system. In the former case, the system should integrate the new reference into the EIS of the matching identity. In the latter case, the system must create a new EIS. This means there will be two inputs to the update process including the existing (prior) EIS created in a previous process and the new references.

Figure 11 shows the general approach to the update configuration for a distributed processing environment that is basically the workflow of Figure 7 with an added input of prior EIS.

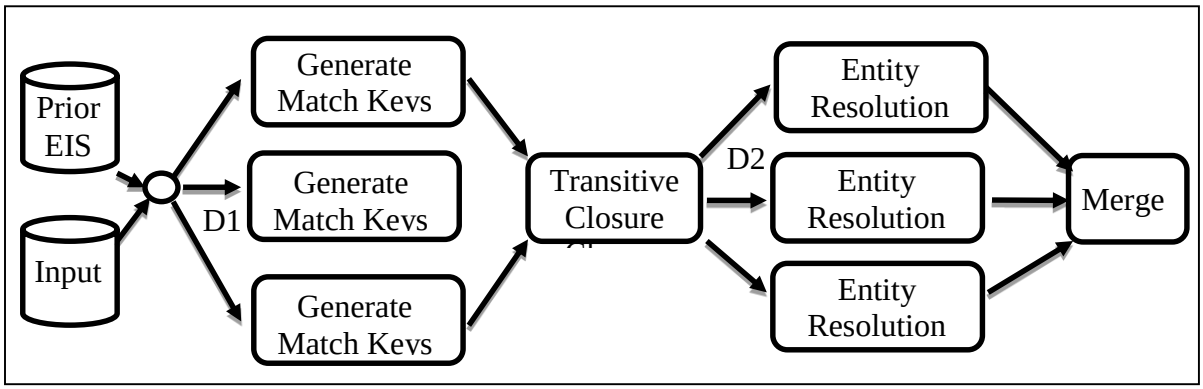


Figure 11: The Basic Workflow for the Update Configuration

Persistent Entity Identifiers

An important goal of EIIM is to create and maintain persistent entity identifiers for each identity under management, i.e. each EIS retains the same identifier from process to process. This is not the case for basic ER systems that do not create and maintain EIS. In these systems the link identifier assigned to a cluster of references at the end of the process is transient because it is not part of a persistent identity structure. The entity identifiers assigned by these systems are only intended to represent the linking (identity) decisions for a particular process acting on a particular input. Typically these system select one survivor record from each link cluster to go forward into further processing steps. After the survivor record has been selected, the link identifier is no longer needed.

The solution to this problem is to treat the prior EIS identifier as a match key in the transitive closure process. To see how this works consider a simple example in which there is only one Prior EIS, one new reference, and a single match key generator G1 as shown in Figure 12

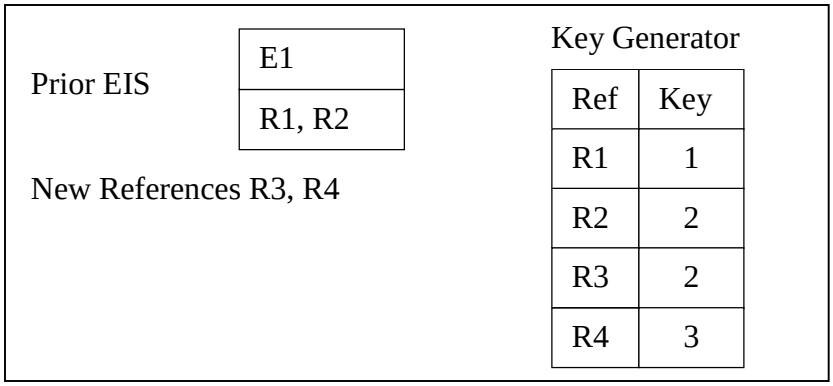


Figure 12: Example Update Scenario

In the scenario of Figure 12 there is a single Prior EIS with Identifier E1 containing two references R1 and R2, two new references R3 and R4, and a single match key generator G1 that produces three distinct match key values 1, 2, and 3. For purposes of the example, assume the generator G1 is aligned with the base match rule, and R2 and R3 match by the base rule. Remember the alignment between the match rule and the generator means if two references match they must generator the same match key, however if they generate the same match key, they do not necessarily match by the rule.

Also, notice in Figure 12 that R1 and R2 do not generate the same match key even though they are in the same EIS. Assuming generator alignment, this means R1 and R2 do not match by the base rule because they generate different match keys. However, it is important to remember EIS are designed to store equivalent references and not all equivalent references match. For example, suppose these are customer references and the base rule is they must match on name and address. The references R1 and R2 may be for the same customer who has changed address, i.e. the address in R1 does not match the address in R2. If the generator G1 is aligned with the name+address match rule, then it is unlikely the hash value produced by the address in R1 will be the same as the hash value produced by the address in R2, consequently the match keys for R1 and R2 will be different.

Nevertheless, these references are equivalent and therefore belong in the same EIS. This EIS may have been formed in a previous automated ER process that used a different base rule, e.g. match on name and telephone number, or these references may have been asserted together in a manual update process.

CONCLUSION

With MDM as with any process where quality is paramount, the fundamental principles of quality management still apply – quality assurance, quality control, and continuous improvement. It all starts with goals and measurement. Without measurement it is impossible to know progress is being made. For MDM the goal is 100% entity identity integrity. The metric is not complicated, but it takes effort to make the measurement using either at truth set or by benchmarking. In addition, there is a limit as to how much accuracy can be obtain only through automation. Where high-precision is required (or desired), a manual, clerical review process must be put into place. To be efficient, the clerical review must be supported by visualization tools, review indicators, and both correction and confirmation assertions.

As in all aspects of data management, Big Data is having an impact. Fortunately, there is a commensurate growth in big data tools, many of which are open source. The large data store paradigm of key-value pairs fits well with the basic architecture of entity resolution and entity identity information management systems. The problems of transitive closure, large components, and update are all solvable, but require a rethinking of traditional entity resolution strategies.

REFERENCES

Baxter, R., Christen, P., Churches, T., 2003. A comparison of fast blocking methods for record linkage. First Workshop on Data Cleaning, Record Linkage, and Object Consolidation. *KDD-2003*, Washington, DC, August 24-27, 2013.

Bianco, G.D., Galante, R., Heuser, C.A., 2011. A fast approach for parallel deduplication on multicore processors. *SAC'11*, March 21-25, 2011, TaiChung, Taiwan.

Christen, P., 2006. A comparison of personal name matching: techniques and practical issues. *Sixth IEEE International Conference on Data Mining Workshops*, pp. 290-294.

Christen, P., 2012. *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer.

Fellegi, I., Sunter, A., 1969. A theory for record linkage. *Journal of the American Statistical Association* 64 (328), 1183-1210.

Isele, R., Jentzsch, A., Bizer, C., 2011. Efficient multidimensional blocking for link discovery without losing recall. *Fourteenth International Workshop on the Web and Databases*. WebDB-2011, June 12, 2011, Athens, Greece.

Kardes, H., Konidena, D., Agarwal, S., Huff, M., Sun, A., 2013. Graph-based approaches for organizational entity resolution in MapReduce. *Proceedings of the TextGraphs-8 Workshop*. October 18, 2013, Seattle, WA, pp. 70-78.

Kirsten, T., Kolb, L., Hartung, M., Gross, A., Kopche, H., Rahm, E., 2010. Data partitioning for parallel entity matching. *Proceedings of the VLDB Endowment*, Vol. 3. No. 2.

Kolb, L., Thor, A., Rahm, E., 2011. Block-based load balancing for entity resolution with MapReduce. *CIKM'11,* October 24-28, 2011, Glasgow, Scotland, pp. 2397-2400.

Maydanchik, A., 2007. *Data Quality Assessment.* Technics Publications.

Papadakis, G., Ioannou, E., Niederée, C., Palpanas, T., Nedjl, W., 2012. *WSDM'12*. February 8-12, 2012, Seattle, WA, pp. 53-62.

Sedgewick, R., Wayne, K., 2011. *Algorithms*, Fourth Edition. Addison Wesley.

Syed, H., Talburt, J.R., Liu, F., Pullen, D., Wu, N., 2012. Developing and refining matching rules for entity resolution. *The 2012 International Conference on Information and Knowledge Engineering* (IKE'12). Las Vegas, Nevada, July 16-29, 2012, pp. 345-350.

Talburt, J., Wang, R., Hess, K., Kuo, E., 2007. An algebraic approach to data quality metrics for entity resolution over large datasets. In: Al-Hakim, L. (Ed.), *Information quality management: Theory and applications*. Idea Group Publishing, Hershey, PA, pp. 1-22.

Talburt, J.R., 2011. *Entity resolution and information quality*. Morgan Kaufmann.

Talburt, J.R., Zhou, Y., 2015. *Entity Information Life Cycle for Big Data: Master Data Management and Information Integration*. Morgan Kaufmann.

Yancey, W., 2007. BigMatch: A program extracting possible matches from a large file. In: *Research Report Series* (Computing #2007-1). Statistical Research Division, U.S. Census Bureau, Washington, DC.

Zhou, Y., Talburt, J., 2011. The role of asserted resolution in entity identity information management. *Proceedings: 2011 Information and Knowledge Engineering Conference* (IKE 2011). Las Vegas, NV, July 18-20, 2011, pp. 291-296.

RECOMMENDED READING

- *Entity Information Life Cycle for Big Data: Master Data Management and Information Integration,* John R. Talburt & Yinle Zhou, 2015, Morgan Kaufmann Publishing

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

John R. Talburt
Black Oak Analytics, Inc.
jtalburt@blackoakgroup.com