QUERY PROCESSING: A FRAMEWORK FOR INTEGRATING ENTITY RESOLUTION

[#1]S.MAMATHA, M.Tech Student,
[#2]V.NEELIMA, Associate Professor,
[#3]Dr.M.SUJATHA, Associate Professor,
Department Of CSE,
JYOTHISHMATHI INSTITUTE OF TECHNOLOGICAL SCIENCES, KARIMNAGAR T.S.INDIA.

ABSTRACT: Entity resolution is the problem of reconciling database references corresponding to the same real-world entities. Given the abundance of publicly available databases that have unresolved entities, we motivate the problem of query-time entity resolution: quick and accurate resolution for answering queries over such 'unclean' databases at query-time. Since collective entity resolution approaches where related references are resolved jointly have been shown to be more accurate than independent attribute-based resolution for off-line entity resolution, we focus on developing new algorithms for collective resolution for answering entity resolution queries at query-time. For this purpose, we first formally show that, for collective resolution, precision and recall for individual entities follow a geometric progression as neighbours at increasing distances are considered. Unfolding this progression leads naturally to a two stage 'expand and resolve' query processing strategy. This paper explores an analysis-aware data cleaning architecture for a large class of SPJ SQL queries. In particular, we propose QuERy, a novel framework for integrating entity resolution (ER) with query processing. The aim of QuERy is to correctly and efficiently answer complex queries issued on top of dirty data. The comprehensive empirical evaluation of the proposed solution demonstrates its significant advantage in terms of efficiency over the traditional techniques for the given problem settings.

Keywords: *entity resolution, relations, query, adaptive.*

I. INTRODUCTION

This paper addresses the problem of analysis-aware data cleaning, wherein the needs of the analysis task dictates which parts of the data should be cleaned. Analysis-aware cleaning is emerging as a new paradigm for data cleaning to support today's increasing demand for (near) real-time analytical applications of big data. Modern enterprises have access to potentially limitless data sources, e.g., web data repositories, social media posts, clickstream data from web portals, etc. Analysts/users usually wish to integrate one or more such data sources (possibly with their own data) to perform joint analysis and decision making. For example, a small store owner may discover an online source (e.g., a web table) containing Amazon's product pricing and may wish to compare that pricing with her own pricing. Several systems have been developed to empower analysts to dynamically discover and merge data sources. For instance, Microsoft Power Query provides features to dynamically find, combine, visualize, share, and query data across a wide variety of online and offline sources. Another example is Trifacta [2], a data transformation platform that employs a predictive interaction framework [15] to enable users to transform raw data into structured formats. However, to the best of our knowledge, such systems have not yet incorporated data cleaning mechanisms.

As a result of merging data from a variety of sources, a given real-world object may often have multiple representations, resulting in data quality challenges. In this paper, we focus on the Entity Resolution (ER) challenge [6,10], the task of which is to discover duplicate entities that refer to the same real-world object and then to group them into a single cluster that uniquely represents that object.

Traditionally, entity resolution, and data cleaning in general, is performed in the context of data warehousing as an offline pre-processing step prior to making data available to analysis – an approach that works well under standard settings. Such an offline strategy, however, is not viable in emerging applications that deal with big data analysis. First, the need for (near) real-time analysis requires modern applications to execute up-to-the-minute analytical tasks, making it impossible for those applications to use time consuming standard back-end cleaning technologies. Another reason is that in the data analysis scenarios that motivate our work, an analyst/user may discover and analyze data as part of a single integrated step. In this case, the system will know "what to clean" only at analysis time (while the user is waiting to analyze the data). Last, given the volume and the velocity of big data, it is often infeasible to expect that one can fully collect or clean data in its entirety. Recent work on analysis-aware data cleaning seeks to overcome the limitations of traditional offline data cleaning techniques. While such solutions address analysis aware data cleaning, they are limited to only simple queries (viz., mention, selection, and/or numerical aggregation queries) executed on top of dirty data. Data analysis, however, often requires a significantly more complex type of queries requiring SQL-style joins. For instance, a user interested in comparative shopping may wish to find

cellphones that are listed on two distinct data sources: Best Buy and Walmart to compare their ratings and reviews. Clearly, the query that corresponds to the user's interest will require joining Best Buy's and Walmart's cellphone-listings. In contrast to our work, the previous approaches cannot exploit the semantics of such a join predicate to reduce cleaning.

Specifically, this paper explores the problem of analysisaware data cleaning for the general case where queries can be complex SQL-style selections and joins spanning single/multiple dirty entity-sets. We propose QuERy, a novel framework for integrating ER with query processing. The objective of QuERy is to efficiently and accurately answer complex SelectProject-Join (SPJ) queries issued on top of dirty data. The predicates in those queries may be associated with any attribute in the entity-sets being queried.

In particular, QuERy leverages the selectivity's offered by the query predicates to reduce the amount of cleaning (by only deduplicating those parts of data that influence the query's answer) and thus, minimizes the total execution time of the query. We propose two variants of QuERy: lazy-QuERy and adaptive-QuERy. The former uses a lazy architecture that attempts to avoid cleaning until it is necessary for the system to proceed. The latter is an adaptive cost-based technique that tries to devise a good plan to decide when to perform cleaning. Both solutions rely on novel polymorphic operators, which are analogous to the common relational algebra operators (i.e., selections and joins) with one exception: they know how to test the query predicates on the dirty data prior to cleaning it. Specifically, these operators utilize sketches of data to perform inexact tests to decide whether parts of dirty data satisfy query predicates.

Overall, the main contributions of this paper are:

- We propose QuERy, a novel framework that integrates ER with query processing to answer complex SQL-style queries issued on top of dirty data (Sections 2 and 4).
- We introduce and formalize the notion of polymorphic operators – a key concept in QuERy (Section 5).
- We develop two different solutions: lazy-QuERy and adaptiveQuERy, which reap the benefits of evaluating the query predicates to minimize the query execution time (Sections 6 and 7).
- We conduct extensive experiments to evaluate the effectiveness of both lazy-QuERy and adaptive-QuERy solutions on real and synthetic datasets (Section 8)

II. RELATED WORK

Entity resolution is a well-recognized problem that has received significant attention in the literature over the past few decades, e.g. [6, 10]. A thorough overview of the existing work in this area can be found in surveys [11, 20]. The majority of previous ER research has focused on improving either its efficiency [16, 21] or quality [5, 7]. With the increasing demand of (near) real-time analytical applications, recent research has begun to consider new ER approaches like analysis-aware ER, progressive ER, incremental ER, etc. Analysis-aware ER. The work on analysis-aware ER has been proposed in [4, 8, 17, 22, 24], of which [4, 24] are the most related to our work. The QDA approach of [4] aims to reduce the number of cleaning steps that are necessary to exactly answer selection queries. It works as follows: given a block B, and a complex selection predicate P, QDA analyzes which entity pairs do not need to be resolved to identify all entities in B that satisfy P. To do so, it models entities in B as a graph and resolves edges belonging to cliques that may change the query answer. To support a selection query, QDA performs vestigiality analysis on each block individually to reduce cleaning steps. QDA is not designed for the larger class of SPJ queries, which is the context of this paper. In contrast, QuERy explores a systematic cost-based approach to jointly optimize both cleaning and query processing over dirty data. It exploits pruning due to both selection and join predicates. It only dictates when a block should be cleaned and is agnostic to how the block is actually cleaned. Thus, it could exploit vestigiality analysis from QDA at the block level to reduce the number of entity pairs that are resolved within a block. In addition, reference [24] is designed to answer aggregate numerical queries over large datasets that cannot be fully cleaned. It focuses on cleaning only a sample of data and utilizing that sample to provide "approximate" answers to aggregate queries. It does not prune cleaning steps due to query predicates. However, QuERy deals with "exact" answers to SPJ queries based on cleaning only the necessary parts of data needed to answer the query. Moreover, not only are the two approaches designed for different types of queries, their motivation is also very different. While [24] is targeting aggregation over very large datasets, QuERy targets applications that perform (near) real-time analysis over dynamic dirty datasets found on the Web. New ER Approaches. Several approaches, e.g., [3, 26], are considering how to clean the data progressively, while interactively analyzing the partially cleaned data to compute better results. Moreover, there has been various incremental cleaning techniques [12,25]. Such techniques address the problem of maintaining an up-to-date ER result when data updates arrive quickly. In addition to such approaches, the ER research community is exploring other novel directions. For example, Data Tamer [23], is an end-to-end data curation system that entails machine learning algorithms with human input to perform schema integration and entity resolution. In addition, NADEEF [9] is a general-purpose data cleaning and repair system that provides appropriate programming abstractions for users to specify data cleaning transformations. The focus of QuERy is thus,

complementary (though different) to that of [23] and [9]. In fact, we envision that QuERy could be useful to these systems to expand their scope to target (near) real-time analysis-aware applications of diverse data sources found on the Web.

III. ENTITY RESOLUTION AND QUERIES: FORMULATION

In this section, we formally introduce the entity resolution problem and also entity resolution queries, and illustrate them using a realistic example — that of resolving authors in a citation database such as CiteSeer or PubMed. In the simplest formulation of the entity resolution problem, we have a collection of references, R = {ri}, with attributes {R.A1, . . . , R.Ak}. Let E = {ej} be the unobserved domain entities. For any particular reference ri , we denote the entity to which it maps as E(ri). We will say that two references ri and rj are co-referent if they correspond to the same entity, E(ri) = E(rj). Note that in the case of an unresolved database, this mapping E(R) is not provided. Further, the domain entities E and even the number of such entities is not known. However, in many domains, we may have additional information about relationships between the references. To model relationships in a generic way, we use

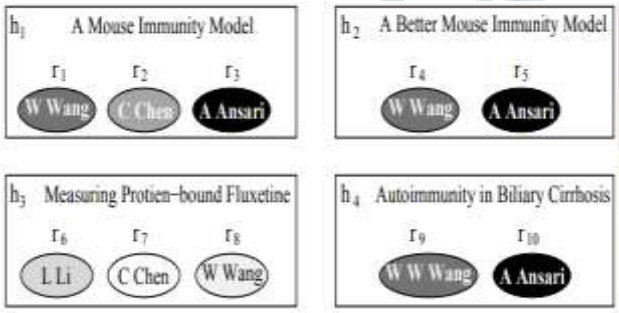

Figure 1: An example set of papers represented as references connected by hyper-edges. References are represented as ovals shaded according to their entities. Each paper is represented as a hyper-edge (shown as a rectangle) spanning multiple references.

a set of hyper-edges H = {hi}. Each hyper-edge connects multiple references. To capture this, we associate a set of references hi .R with each hyper-edge hi . Note that each reference may be associated with zero or more hyper-edges.

Let us now look at a sample domain to see how it can be represented in our framework. Consider a database of academic publications similar to DBLP, CiteSeer or PubMed. Each publication in the database has a set of author names. For every author name, we have a reference ri in R. For any reference ri , ri .N ame records the observed name of the author in the publication. In addition, we can have attributes such as R.Email to record other information for each author reference that may be available in the paper. Now we come to the relationships for this domain. All the author references in any publication are connected to each other by a co-author relationship. This can be represented using a hyper-edge hi ∈ H for each publication and by having rj ∈ hi .R for each reference rj in the publication. If publications have additional information such as title, keywords, etc, they are represented as attributes of H.

To illustrate, consider the following four papers, which we will use as a running example:

1. W. Wang, C. Chen, A. Ansari, “A mouse immunity model”
2. W. Wang, A. Ansari, “A better mouse immunity model”
3. L. Li, C. Chen, W. Wang,“Measuring protein-bound fluxetine”
4. W. W. Wang, A. Ansari, “Autoimmunity in biliary cirrhosis”

To represent them in our notation, we have 10 references {r1, . . . , r10} in R, one for each author name, such that r1.N ame = ‘W Wang’, etc. We also have 4 hyper-edges {h1, . . . , h4} in H, one for each paper. The first hyper-edge h1 connects the three references r1, r2 and r3 corresponding to the names ‘W. Wang’ , ‘C. Chen’ and ‘A. Ansari’. This is represented pictorially in Figure 1.

Given this representation, the entity resolution task is defined as the partitioning or clustering of the references according to the underlying entity-reference mapping E(R). Two references ri and rj should be assigned to the same cluster if and only if they are coreferent, i.e., E(ri) = E(rj). To illustrate, assume that we have six underlying entities for our example. This is illustrated in Figure 1 using a different shading for each entity. For example, the ‘Wang’s of papers 1, 2 and 4 are names of the same individual but the ’Wang’ from paper 3 is a reference to a different person. Also, the ‘Chen’s from papers 1 and 3 are different individuals. Then, the correct entity resolution for our example database with 10 references returns 6 entity clusters: {{r1, r4, r9}, {r8}, {r2}, {r7}, {r3, r5, r10}, {r6}}. The first two clusters correspond to two different people named ‘Wang’, the next two to two different people named ‘Chen’, the fifth to ‘Ansari’ and the last to ‘Li’.

Any query to a database of references is called an entity resolution query if answering it requires knowledge of the underlying entity mapping E(R). We consider two different types of entity resolution queries. Most commonly, queries are specified using a particular value a for an attribute R.A of the references that serves as a ‘quasi-identifier’ for the underlying entities. Then the answer to the query Q(R.A = a) should partition or group all references that have r.A = a according to their underlying entities. For references to people, the name often serves as a weak or noisy identifier. For our example bibliographic domain, we consider queries specified using R.N ame. To retrieve all papers written by some person named ‘W. Wang’, we issue a query using R.N ame and ‘W. Wang’. Since names are ambiguous, treating them as identifiers leads to undesirable results. In this case, it would be incorrect to return the set {r1, r4, r8} of all references with name ‘W Wang’ as the answer to our query. This answer does not indicate that r8 is not the same person as the other two. Additionally, the answer should include the

reference r9 for 'W W Wang', that maps to the same entity as the author of the first paper. Therefore, the correct answer to the entity resolution query on 'W Wang' should be the partition {{r1, r4, r9}, {r8}}.

Entity resolution queries may alternatively be specified using a specific reference. Imagine a CiteSeer user looking at a paper that contains some author name. The user may be interested in looking up other papers written by the same author, even though they may not know who that author is precisely. Then the correct answer to a query on the reference r is the group of references that are coreferent to r, or, in other words, correspond to the same underlying entity. In our example, consider a query specified using the reference r1 corresponding to the name 'W. Wang' in the first paper. Then the correct answer to the query is the set of references {r1, r4, r9}. To distinguish it from the first type of entity resolution query, note that it does not include the cluster {r8} corresponding to the other entity that also has name 'W. Wang'. This second query type may be answered by first reducing it to an instance of the first type as Q(R.A = r1.A), and then selecting the entity corresponding to reference r1. We denote this as $\sigma_{E(R)=E(r1)}(Q(R.A = r1.A))$. In the rest of this paper, we focus only on queries of the first type.

IV. COLLECTIVE ENTITY RESOLUTION AND RELATIONAL CLUSTERING

Although entity resolution for queries has not been studied in the literature, the general entity resolution problem has received a lot of attention. We review related work in detail in Section 8. In this section, we briefly review the different categories of proposed approaches before discussing how they may be adapted for query-time entity resolution.

In most entity resolution applications, data labeled with the underlying entities is hard to acquire. Our focus is on unsupervised approaches for resolving entities. Traditionally, attributes of individual references, such as names, affiliation, etc., for person references, are used for comparing references. A similarity measure is generally employed over attributes, and only those pairs of references that have attribute similarity above a certain threshold are considered to be co-referent. This attribute-based entity resolution approach (A) often runs into problems. In our example, it is hard to infer with just attributes that references r1 and r8 are not co-referent although they have the same name, while r1 and r9 are co-referent although their names are different.

When relations between references are available, they may also be taken into account for computing similarities in the naive relational entity resolution approach (NR) (Ananthakrishna et al., 2002; Bhattacharya & Getoor, 2007). For computing similarities between two references, this approach additionally considers the attributes of the related references when comparing the attributes of their related references. In our example, this approach returns a higher similarity between r1 ('W. Wang') and r9 ('W. W. Wang') than the attribute-based approach, since they have co-authors r3 and r10 with very similar (identical, in this case) names. Although this approach can improve performance in some cases, it does not always work. For instance, the two 'W. Wang' references r1 and r8 are not co-referent, though they both have co-authors with identical names 'C. Chen'.

Instead of considering the attribute similarities of the related references, the collective entity resolution approach (Pasula et al., 2003; Bhattacharya & Getoor, 2004; Singla & Domingos, 2004; McCallum & Wellner, 2004; Li, Morie, & Roth, 2005; Dong et al., 2005; Kalashnikov et al., 2005) takes into account the resolution decisions for them. In our previous example, the correct evidence to use for the pair of references r1 and r8 is that their co-author references do not map to the same entity, although they have similar names. Therefore, in order to resolve the 'W. Wang' references in the collective resolution approach, it is necessary to resolve the 'C. Chen' references as well, instead of considering the similarity of their attributes. The collective entity resolution approach has recently been shown to improve entity resolution accuracy over the previous approaches but is computationally more challenging. The references cannot be resolved independently. Instead, any resolution decision is affected by other resolutions through hyper-edges.

In earlier work (Bhattacharya & Getoor, 2004, 2006, 2007), we developed a relational clustering algorithm (RC-ER) for collective entity resolution using relationships. The goal of this approach is to cluster the references according to their entities taking the relationships into account. We associate a cluster label r.C with each reference to denote its current cluster membership. Starting from an initial set of clusters C = {ci} of references, the algorithm iteratively merges the pair of clusters that are the most similar. To capture the collective nature of the cluster assignment, the similarity measure between pairs of clusters considers the cluster labels of the related references. The similarity of two clusters ci and cj is defined as a linear combination of their attribute similarity simA and their relational similarity sim_R:

$$sim(c_i, c_j) = (1 - \alpha) \times sim_A(c_i, c_j) + \alpha \times sim_R(c_i, c_j) \quad (1)$$

where α ($0 \leq \alpha \leq 1$) is the combination weight. The interesting aspect of the collective approach is the dynamic nature of the relational similarity. The similarity between two references depends on the current cluster labels of their related references, and therefore changes when related references change clusters. In our example, the similarity of the two clusters containing references 'W. Wang' and 'W. W. Wang' increases once their co-author references named 'A. Ansari' are assigned to the same cluster. We now briefly review how the two components of the similarity measure are defined.

Attribute Similarity: For each reference attribute, we use a similarity measure that returns a value between 0 and 1 for two attribute values indicating the degree of similarity between them. Several sophisticated similarity measures

have been developed for names, and popular TF-IDF schemes may be used for other textual attributes such as keywords. The measure that works best for each attribute may be chosen. Finally, a weighted linear combination of the similarities over the different attributes yields the combined attribute similarity between two reference clusters.

Relational Similarity: Relational similarity between two clusters considers the similarity of their 'cluster neighborhoods'. The neighborhood of each cluster is defined by the hyper-edges associated with the references in that cluster. Recall that each reference r is associated with one or more hyper-edges in H. Therefore, the hyper-edge set c.H for a cluster c of references is defined as

$$c.H = \bigcup_{r \in \mathcal{R} \wedge r.C = c} \{h \mid h \in \mathcal{H} \wedge r \in h.R\} \quad (2)$$

This set defines the hyper-edges that connect a cluster c to other clusters, and are the ones that relational similarity needs to consider. To illustrate, when all the references in our running example have been correctly clustered as in Figure 1(b), the hyper-edge set for the larger 'Wang' cluster is {h1, h2, h4}, which are the hyper-edges associated with the references r1, r4 and r9 in that cluster.

Given the hyper-edge set for any cluster c, the neighborhood N br(c) of that cluster c is the set of clusters labels of the references spanned by these hyper-edges:

$$Nbr(c) = \bigcup_{h \in c.H, r \in h} \{c_j \mid c_j = r.C\} \quad (3)$$

For our example 'Wang' cluster, its neighborhood consists of the 'Ansari' cluster and one of the 'Chen' clusters, which are connected by its edge-set. Then, the relational similarity measure between two clusters, considers the similarity of their cluster neighborhoods. The neighborhoods are essentially sets (or multi-sets) of cluster labels and there are many possible ways to define the similarity of two neighborhoods (Bhattacharya & Getoor, 2007). The specific similarity measure that we use for our experiments in this paper is Jaccard similarity[1] :

$$sim_R(c_i, c_j) = Jaccard(Nbr(c_i), Nbr(c_j)) \quad (4)$$

Clustering Algorithm: Given the similarity measure for a pair of clusters, a greedy relational clustering algorithm can be used for collective entity resolution. Figure 2 shows high-level pseudo-code for the complete algorithm. The algorithm first identifies the candidate set of potential duplicates using a 'blocking' approach (Hern´andez & Stolfo, 1995; Monge & Elkan, 1997; McCallum, Nigam, & Ungar, 2000). Next, it initializes the clusters

Algorithm RC-ER (Reference set R)
1. Find similar references in R using blocking
2. Initialize clusters using bootstrapping
3. For clusters ci, cj such that similar(ci, cj)
4. Insert hsim(ci, cj), cj , cj i into priority queue
5. While priority queue not empty
6. Extract hsim(ci, cj), ci, cj i from queue
7. If sim(ci, cj) less than threshold, then stop
8. Merge ci and cj to new cluster cij
9. Remove entries for ci and cj from queue
10. For each cluster ck such that similar(cij , ck)
11. Insert hsim(cij , ck), cij , cki into queue
12. For each cluster cn neighbor of cij
13. For ck such that similar(ck, cn)
14. Update sim(ck, cn) in queue

Figure 2: High-level description of the relational clustering algorithm

of references, identifies the 'similar' clusters — or potential merge-candidates — for each cluster, inserts all the merge-candidates into a priority queue and then iterates over the following steps. At each step, it identifies the current 'closest pair' of clusters from the candidate set and merges them to create a new cluster. It identifies new candidate pairs and updates the similarity measures for the 'related' cluster pairs. This is the key step where evidence flows from one resolution decision to other related ones and this distinguishes relational clustering from traditional clustering approaches. The algorithm terminates when the similarity for the closest pair falls below a threshold or when the list of potential candidates is exhausted. The algorithm is efficiently implemented to run in $O(nk \log n)$ time for n references where each 'block' of similar names is connected to k other blocks through the hyper-edges.

3.1 Issues with Collective Resolution for Queries

In previous work, we (and others) have shown that collective resolution using relationships improves entity resolution accuracy significantly for offline cleaning of databases. So, naturally, we would like to use the same approach for query-time entity resolution as well. However, while the attribute-based and naive relational approaches discussed earlier can be applied at query-time in a straight-forward fashion, that is not the case for collective resolution. Two issues come up when using collective resolution for queries. First, the set of references that influence the resolution decisions for a query need to be identified. When answering a resolution query for 'S. Russell' using the attribute-based approach, it is sufficient to consider all papers that have 'S. Russell' (or, similar names) as author name. For collective resolution, in contrast, the co-authors of these author names, such as 'P.

Norvig' and 'Peter Norvig', also need to be clustered according to their entities. This in turn requires clustering their co-authors and so on. So the first task is to analyze these dependencies for collective resolution and identify the references in the database that are relevant for answering a query. But this is not enough. The set of references influencing a query may be extremely large, but the query still needs to be answered quickly even though the answer may not be completely accurate. So the second issue is performing the resolution task at query-time. These are the two problems that we address in the next few sections.

V. ADAPTIVE-QUERY SOLUTION

The previous solution is considered lazy since it tries to delay the cleaning of dirty entities as much as possible. While such an approach will reduce the cost of cleaning, it might increase the cost of processing the query. For example, assume that all sketches end up reaching the deduplicate operator (viz., the topmost operator in Figure 11), meaning that their corresponding blocks need to be cleaned. In this case, the time spent in trying to filter away these blocks is wasted. In fact, cleaning these blocks eagerly (without passing their sketches up the tree) might be more efficient. To address this issue, we implement a different solution which is an adaptive cost-based approach that, given a query tree (with polymorphic operators) and dirty entity-sets, can devise a good plan to simultaneously clean and process the query. The key intuition hinges on placing decision nodes as the bottommost nodes in the query tree, as presented in Figure 12. The task of such decision nodes is to decide if eagerly cleaning some dirty blocks is more efficient (in terms of the overall query execution time) than delaying their cleansing until the last stage as in the lazy solution. The conjecture of placing these nodes at the bottom is to allow adaptive QuERy to make the "cleaning a block eagerly versus passing it up the tree" decision, from the start of query execution time. Note our adaptive solution is general, and it does not require such nodes to be placed at the bottom. Our adaptive cost-based solution consists of two steps. In the first step, we use a sampling technique to collect different statistics (e.g., selectivity of predicates, cost of join, etc.). In the second step, the decision nodes utilize these statistics to make their smart decisions.

VI.CONCLUSIONS

In this paper, we have studied the problem of analysis aware data cleaning. We have developed QuERy, a novel architecture for integrating ER with query processing to answer complex SQL-like queries issued on top of dirty data. We empirically showed how our approach is significantly better compared to cleaning the entire dataset, especially when the query is very selective.

We then proposed a two-stage 'expand and resolve' strategy for answering queries based on this analysis, using two novel expansion operators. We showed using our analysis that it is sufficient to consider neighbors up to small expansion depths, since resolution accuracy for the query converges quickly with increasing expansion level. The second challenge for answering queries is that the computation has to be quick.

REFERENCES

[1] E. Ioannou, W. Nejdl, C. Nieder´ee, and Y. Velegrakis. On-the-fly entity-aware query processing in the presence of linkage. VLDB, 2010.

[2] Ananthakrishna, R., Chaudhuri, S., & Ganti, V. (2002). Eliminating fuzzy duplicates in data warehouses. In The International Conference on Very Large Databases (VLDB), Hong Kong, China.

[3] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. VLDB, 2014.

[4] H. Altwaijry, D. V. Kalashnikov, and S. Mehrotra. Query-driven approach to entity resolution. VLDB, 2013.

[5] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In VLDB, 2002.

[6] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. VLDB J., 2009.

[7] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. TKDD, 2007.

[8] I. Bhattacharya and L. Getoor. Query-time entity resolution. JAIR, 2007.

[9] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. Nadeef: a commodity data cleaning system. In SIGMOD, 2013.

[10] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In SIGMOD, 2005.

[11] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. TKDE, 2007.

[12] A. Gruenheid, X. L. Dong, and D. Srivastava. Incremental record linkage. VLDB, 2014.

[13] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation. In SIGMOD Record, 1999.

[14] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller. Framework for evaluating clustering algorithms in duplicate detection. VLDB, 2009.

[15] J. Heer, J. M. Hellerstein, and S. Kandel. Predictive interaction for data transformation.