Preface

The ability to detect and correct errors in the data, and more broadly to develop techniques for data quality assessment, has long been recognized as critical to the functionality of a large number of applications, in areas ranging from business management to data-intensive science. While many of the technical issues associated with data quality have been known for quite some time, novel applications still pose original challenges, while advances in data management technology offer ideas for novel approaches.

The sixth in a workshop series dedicated specifically to problems of Quality in Databases, QDB'08 is a qualified forum for presenting and discussing novel ideas and solutions related to the problems of assessing, monitoring, improving, and maintaining the quality of data. Previous editions of the workshop were co-located with top-level data management conferences, namely SIGMOD and VLDB.

The workshop on Management of Uncertain Data (MUD) is the third in a row addressing the area of techniques for handling uncertainty in data. The first workshop took place at the University of Twente in 2006 in the Twente Data Management workshop series and last year the workshop was also co-located with VLDB.

This year, QDB and MUD have joined forces and have been able to offer a rich and qualified program, consisting of 12 original research papers, each subject to the scrutiny of at least three reviewers, and an invited talk. The workshop structure reflects three main areas of interest from the community. The first session addresses traditional issues of Record Linkage and Data Correction using distinctly novel techniques. The second session is entirely dedicated to extending database operations to handle uncertainty in data. Finally, original methods and techniques for Data Quality assessment are presented in the third session.

The workshop includes an invited talk by Zachary Ives, of University of Pennsylvania, addressing the consistency issues that are faced in the context of the Orchestra project, when multiple independent evolution paths in the database are supported.

We would like to thank the PC members for their effort in reviewing the papers and of course the authors of all submitted papers for their work. We also would like to thank the Centre for Telematics and Information Technology (CTIT) for sponsoring the proceedings. Last, but not least, we would like to thank the VLDB organizers for their support in organizing this workshop.

Ander de Keijzer
Maurice van Keulen
Paolo Missier
Xuemin Lin

Program Committee Workshop on Management of Uncertain Data

Patrick Bosc, IRISA/ENSSAT - University of Rennes 1
Nilesh Dalvi, Yahoo! Research
Alex Dekhtyar, California Polytechnic State University
Maarten Fokkinga, University of Twente
Ander de Keijzer, University of Twente
Maurice van Keulen, University of Twente
Thomas Lukasiewicz, Oxford University
Gabriella Pasi, Università degli Studi di Milano Bicocca, Milano
Olivier Pivert, IRISA/ENSSAT
Sunil Prabhakar, Purdue University
Dan Suciu, University of Washington and Microsoft Research
Martin Theobald, Stanford University
Guy De Tré, Ghent University
Jef Wijsen, University of Mons-Hainaut
Vladimir Zadorozhny, University of Pittsburgh

Program Committee Workshop on Quality in Databases

Arvind Arasu, Microsoft, USA
Carlo Batini, University of Milano Bicocca, Italy
Laure Berti, IRISA, France
Tiziana Catarci, Universita' di Roma La Sapienza, Italy
Ahmed K. Elmagarmid, Purdue University, USA
Suzanne Embury, University of Manchester, UK
Alvaro Fernandes, University of Manchester, UK
Helena Galhardas, Technical University of Lisbon, Portugal
Michael Gertz, UC Davis, USA
Mauricio Hernandez, IBM Almaden, USA
Vipul Kashyap, Partners HealthCare System, USA
Raghav Kaushik, Microsoft Research, USA
Chen Li, UC Irvine, USA
Andrea Maurino, Milano Bicocca, Italy
Felix Naumann, Hasso-Plattner-Institute, Germany
Monica Scannapieco, ISTAT, Italy
Divesh Srivastava, AT&T-Research, USA
Vassilios Verykios, Univ. of Thessaly, Volos, Greece
Wei Wang, UNSW, Australia

Table of Contents

Orchestra: Sharing Inconsistent Data in a Consistent Way

Zachary Ives

One of the most pressing needs in business, government, and science is to bring together structured data from a variety of systems, formats, and terminologies. For instance, the emerging field of systems biology seeks to unify biological data to get a big-picture view of the processes within living organisms. Many organizations have set up databases designed to be "clearing houses" for specific types of information: each is separately maintained, cleaned, and curated, and has its own schema and terminology. Updates are constantly made as hypothesized relationships are confirmed or refuted, or new discoveries are made. The different databases contain complementary information that must be integrated to get a complete picture - and each database may have data of different quality or relevance to a domain. However, there is often no consensus on what the definitive answers are - each site may have different beliefs.

The Orchestra project focuses on how to support exchange of data (and updates) among collaborators with evolving databases, in a way that accommodates disagreement, different schemas, and different levels of authority and quality. Orchestra considers collaborators' databases to be logical *peers* into which data can be imported and then locally modified. It allows for a network of *schema mappings* that interrelate peers, annotated with *trust policies* specifying the conditions under which a peer is willing to import data. As a data item is mapped from site to site in the system, its *provenance* is recorded; a peer's trust policies use this provenance (and the values of the data) to assign a score to each incoming data item (based on perceived quality or relevance), and the peer then uses this score to reconcile conflicts and compute a consistent data instance, whose contents may be unique to the peer. The scores assigned to the individual sources can even be *learned* based on user feedback about query answers. The end result is a system that allows each database to selectively diverge from the others as appropriate, but to remain "in sync" in all other cases.

Joint work with Todd J. Green, Grigoris Karvounarakis, Nicholas Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, Muhammad Salman Mehmood, Koby Crammer, Fernando Pereira, and Sudipto Guha.

Training Selection for Tuning Entity Matching

Hanna Köpcke
University of Leipzig, Germany
koepcke@informatik.uni-leipzig.de

Erhard Rahm
University of Leipzig, Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

Entity matching is a crucial and difficult task for data integration. An effective solution strategy typically has to combine several techniques and to find suitable settings for critical configuration parameters such as similarity thresholds. Supervised (training-based) approaches promise to reduce the manual work for determining (learning) effective strategies for entity matching. However, they critically depend on training data selection which is a difficult problem that has so far mostly been addressed manually by human experts. In this paper we propose a training-based framework called STEM for entity matching and present different generic methods for automatically selecting training data to combine and configure several matching techniques. We evaluate the proposed methods for different match tasks and small- and medium-sized training sets.

1. INTRODUCTION

Entity matching (also known as entity resolution, deduplication, record linkage, object matching, fuzzy matching, similarity join processing, reference reconciliation) is the task of identifying object instances or entities referring to the same real-world object. It is a crucial step in data cleaning and data integration. Entities to be resolved may reside in distributed, typically heterogeneous data sources or may be stored in a single data source, e.g., in a database or search engine store. They may be physically materialized or dynamically be requested from sources, e.g., by database queries or keyword searches.

Entities from web data sources are particularly challenging to match as they are often highly heterogeneous and of limited data quality, e.g., regarding completeness and consistency of their descriptions. Fig. 1 illustrates some of the problems for the popular web entity search engine Google Scholar and five duplicate entries for the same paper. The bibliographic entities have automatically been extracted from web pages or PDF documents and contain numerous quality problems such as misspelled author names, different ordering of authors, heterogeneous venue denominations etc.. Google Scholar performs already an entity matching by clustering references to the same publication to aggregate their citations and fulltext sources. However as the duplicates in the example show, the obtained results are far from perfect influenced by the mentioned quality and heterogeneity problems. This illustrates that there is a big potential for improving data quality by better entity matching techniques. This would be critical for tasks requiring the examination of all duplicates, e.g., to collect all citations of publications for a citation analysis. Similar match problems appear in many application domains, e.g., to compare prices for equivalent products (DVDs, cameras, hotel rooms, etc.) offered in the same or different web sites.

Numerous approaches have been proposed for entity matching (EM) especially for structured data, e.g., in relational databases [13]. Due to the large variety of data sources and entities to match there is no single "best" solution. Instead it is often beneficial to combine several algorithms for improved EM quality, e.g., to consider the similarity of several attributes or related entities. The need to support multiple approaches has led to the development of several entity matching frameworks, e.g., [12], [27]. Such frameworks allow the flexible combination and customization of different methods to achieve good quality results for a given EM task. Unfortunately, the flexibility comes at the price of requiring the user to determine a suitable match strategy for a particular problem. Key decisions to be made include

- Selecting the attributes to be used for matching,
- Choosing a similarity function to be applied, and
- Selecting a threshold for the similarity values above which entities are considered to match.

The chosen configuration can have a large impact on the overall quality but even experts will find it difficult and time-consuming to determine a good selection. This is because there are typically numerous possibilities for each decision resulting in a huge number of possible combinations from which to choose for configuring an entity matching strategy. We thus see a strong need for self-tuning entity matching to largely automatically determine an effective EM strategy for a given task.

While a number of previous studies has investigated training-based (supervised) machine learning for entity matching (see related work), the challenge of self-tuning entity matching is far from solved. An important issue is how training data is selected. Most previous studies on training-based entity matching have provided little details on the selection and size of training data and have not studied the influence of different training sets [5], [9]. In other cases, the authors used a relatively large amount of training data thus favoring good match quality however at the expense of a high manual overhead for labeling [8], [21], [22]. In this study we propose different generic methods for automatically selecting training data to be labeled. The training selection methods are part of a new generic framework that supports the automatic construction of entity matching strategies. In this study the proposed framework serves as an evaluation platform to compare

VLDB '08, August 24-30, 2008, Auckland, New Zealand.

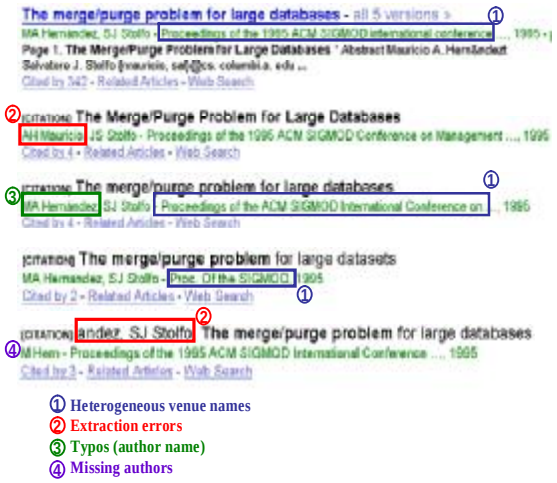


Figure 1: Duplicate paper entities in Google Scholar

different training selection methods and determine the impact of different training sizes on EM accuracy.

The main contributions of this paper are:

- Presentation of a comprehensive, training-based framework for automatically constructing (learning) entity matching strategies called STEM for **s**elf-**t**uning **e**ntity **m**atching
- Description of methods to automatically generate training data for entity matching.
- Comparative evaluation of the training selection methods for different training sizes and four match tasks of different application domains. We specifically analyze the effectiveness for small training sizes which incur only a modest effort for labeling.

In the next section we outline our framework for tuning entity matching. We then propose two methods for selection of training data. Section 4 provides a comprehensive evaluation of the effectiveness of the methods for four match tasks. Related work is reviewed in Section 5 before we conclude.

2. STEM: A TRAINING-BASED ENTITY MATCHING FRAMEWORK

2.1 Problem definition

We assume the sets of entities to be resolved come from two sources S_A and S_B. Entities are of a particular semantic entity type (e.g., publication or product). Each entity is represented by a set of attribute values. The entity matching problem is to find all pairs of equivalent entities which refer to same real-world object.

Definition 1 (Entity match problem / match task): Entity matching is a process which takes as input two sets of entities $A \subseteq S_A$ and $B \subseteq S_B$ of a particular semantic entity type from data sources S_A and S_B. The output is a subset of A x B containing all correspondences between entities representing the same real-world object.

Note that the inputs need not be entire sources but can be subsets of them, e.g., the results of queries. Resolving entities within a single source is supported as a special case ($A=B$, $S_A = S_B$).

To solve an entity matching problem we want to utilize several matchers in combination. A *matcher* is defined as follows:

Definition 2 (Matcher): A matcher m takes as input two sets of entities $A \subseteq S_A$ and $B \subseteq S_B$ of a particular semantic entity type from data sources S_A and S_B. It produces as output a similarity table $T = \{(a, b, s) \mid a \in A, b \in B, s \in [0,1]\}$. The similarity value s indicates the strength of the similarity between two entities $a \in A$ and $b \in B$.

Given a match task and a set of matchers the general objective of our framework is to automatically find an optimal entity matching strategy. An EM strategy defines a selection, configuration and combination of matchers to solve an entity matching problem. It can be defined as follows:

Definition 3 (EM strategy): An EM strategy (S_A, S_B, M, f) for two data sources S_A and S_B of a particular semantic entity type consists of a set M of matchers and a decision function f. For any entity pair (a, b| $a \in S_A$, $b \in S_B$), f applies the matchers M and assigns a label $l \in \{match, non\text{-}match\}$. For any subsets $A \subseteq S_A$ and $B \subseteq S_B$, the EM strategy returns as output a set of correspondences $C = \{(a, b| a \in A, b \in B, f(a, b) = match)\}$ consisting of all entity pairs labeled with l = match by f.

The general tuning problem can be defined as follows:

Definition 4 (General tuning problem): Given two sets of entities $A \subseteq S_A$ and $B \subseteq S_B$ of a particular semantic entity type from data sources S_A and S_B, the general tuning problem is to find an optimal EM strategy s which maximizes a utility function U.*

The utility function U may take diverse factors into account, in particular matching accuracy and execution time. For the initial evaluation of our framework we focus on the matching accuracy measured in terms of precision, recall and F-measure (Section 4.1). We plan to consider more general utility functions including execution time as future work.

2.2 Framework architecture

Fig. 2 illustrates the architecture and use of our generic entity matching framework STEM. With generic we mean that the framework should be applicable to different application domains, to different data sources and to different subsets of a data source.

We distinguish two main processing phases or modes: 1) the specification and 2) the application of an EM strategy. The *specification phase* determines suitable strategies to solve a new EM problem. This phase is typically determined offline, either completely manually or largely automatically by machine learning techniques based on training data. EM strategies determined in the specification phase are stored in a *repository* together with describing properties (input sources, entity type, output characteristics, observed execution times, etc.).

In the *application phase* an EM strategy is selected from the repository and applied to the input entities to be resolved. This phase may be performed offline or online, e.g., for input data that is derived at runtime by queries or other application processing (e.g., mashups). The result of applying the EM strategy is a set of

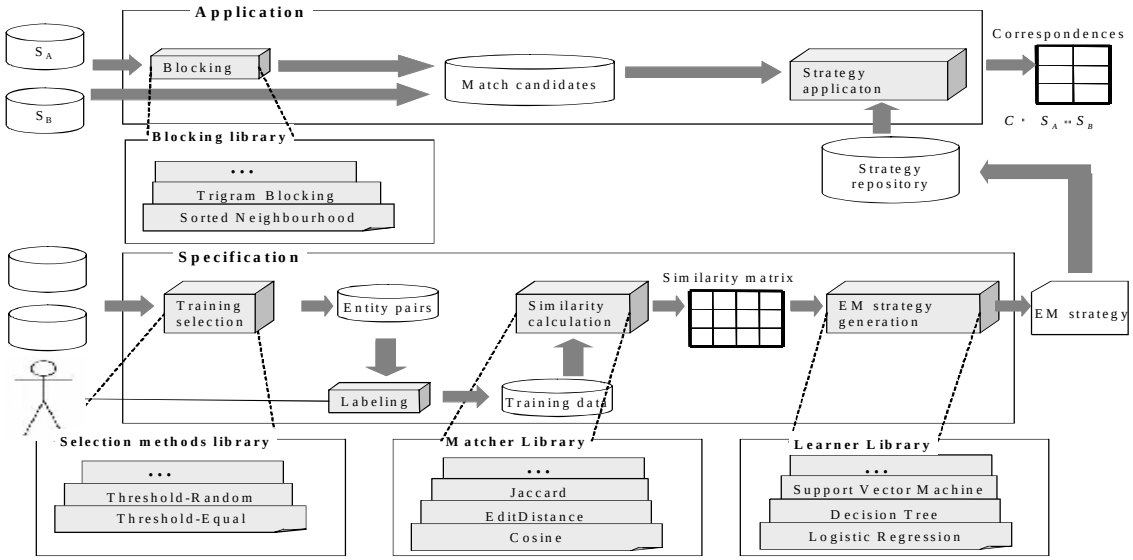


Figure 2: Overview of STEM

correspondences indicating matching entities, i.e. entities which are considered to represent the same real-world object. The EM strategy may be applied on the whole input data or on the result of a blocking step. Blocking is needed for large inputs and reduces the search space for entity matching from the Cartesian product S_A x S_B to a small subset of the most likely correspondences. There are many known algorithms for blocking [3] and we provide several of them, e.g., to exclude all pairs with very low string similarity.

The selection of an EM strategy is mainly determined by the given input sources; in case of multiple choices the selection may be controlled manually or based on statistical data from previous use cases such as execution time or result quality. In this paper, we manually specify the EM strategy to be applied since we want to comparatively evaluate different strategies. In future work we plan to automatically select from predefined EM strategies as an additional kind of tuning. For example, this feature can be useful to consider runtime constraints to select a fast EM strategy with sufficient recall/precision.

In this paper we focus on the training-based generation of EM strategies in the specification phase. This is performed offline because it requires some manual guidance for labeling training entities resulting in a semi-automatic generation of EM strategies. The automatically executed steps can also be time-consuming especially when we generate several strategies to choose from in the application phase. The workflow of determining an EM strategy is illustrated in the lower part of Fig. 2. It consists of three main steps:

- *Generation of training data.* We do not require the manual specification of training data but allow that training data is automatically selected from the entity sets to be matched or other data sets from the same application domain. The result of this step is a set of entity pairs T and a labeling. The labeling is determined manually and indicates for each pair whether the two entities match or not. The pairs to be labeled are determined based on a training selection method.
- *Similarity computation.* Multiple matcher algorithms are applied to determine the similarity for entity pairs in the training data T. Either all supported matchers or a specified subset of them is applied to the training input. The result of this step is a similarity matrix indicating a similarity value for each training pair and matcher.
- *Learning.* The final step is the adoption of a supervised learner algorithm to determine the EM strategy. The learner uses the similarity matrix and the manual labeling as input. It determines the execution order and weighting of the matchers and specifies how matchers are combined.

For each of the three steps our framework provides several methods to choose from as well as additional configuration options. In particular we support several approaches for training selection, for matching, and for learning. The approaches used in our evaluation are (partially) indicated in Fig. 2. The modularity of our framework allows adding further methods if necessary or promising.

The methods for training selection are presented in the next section. The matcher library provides different similarity functions, in particular generic string similarity measures (edit distance, q-grams, cosine, TF/IDF, etc), for defining attribute matchers. An attribute matcher uses one of the similarity functions and applies them to a pair of corresponding attributes of the input entity sets for which the similarity is to be computed. Additional matchers utilizing context information or auxiliary information such as dictionaries can be added as needed.

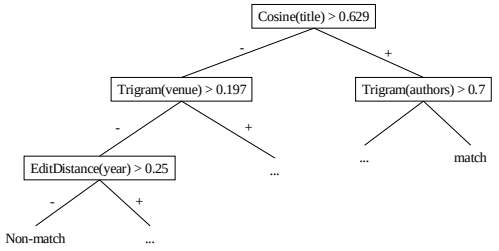


Figure 3: Sample EM strategy for a bibliographic match task generated by a decision tree learner

When all (attribute) matchers should be considered for determining an EM strategy all applicable similarity functions are employed for each attribute pair. We may also use a predefined subset of the matchers to speed up the execution time for matching both during the specification and application phase. Moreover, a smaller number of matchers may require fewer training data and still achieve sufficient EM quality.

The learner library provides several supervised machine learning algorithms or learners, in particular decision tree, SVM (support vector machine), logistic regression and their combination. The learners treat the task of determining an EM strategy as a two-class (match or non-match) classification problem. On the basis of the similarity values for the training examples the learners automatically determine the order and weighting of the matchers and specify how they are combined. Fig. 3 shows a sample EM strategy for a bibliographic match task as determined by the decision tree learner. Each node of the decision tree contains a test whether or not a certain similarity threshold is exceeded for a selected matcher. The match decision is reached by starting with the test of the root node and continuing with the further tests until a leaf node is reached.

While the flexibility of the architecture allows the generation of many strategies for different training selection methods, matcher configurations, and learners, we will here use this flexibility mainly for the comparative evaluation of different approaches. The goal is to identify effective default methods so that only a few EM strategies need to be generated for selection in the application phase.

3. TRAINING DATA SELECTION

The effectiveness of supervised machine learning methods critically depends on the size and quality of the available training data. For entity matching it is important that the training data is representative for the entities to be matched and exhibit the variety and distribution of errors observed in practice. Furthermore, the training data should allow the observation of differences between the available matcher algorithms so that an effective combination of different algorithms can be learned. This requires that the training data contain a sufficient number of both matching as well as non-matching entity pairs. On the other hand, for tuning entity matching it is important to limit the manual overhead for labeling. Hence, we wish to keep the number of training entity pairs to be labeled manually as low as possible.

Most previous studies on training-based entity matching have provided little details on the selection and size of training data and have not studied the influence of different training sets [5], [9]. In other cases, the authors used a relatively large amount of training data thus favoring good match quality however at the expense of a high manual overhead for labeling [8], [21], [22]. In this study we propose different generic methods for automatically selecting training data to be labeled. Furthermore, we want to use the flexibility of our framework to evaluate these methods and determine the impact of different training sizes on EM accuracy.

A naïve method for training selection is to randomly choose entities from the sets of entities to be resolved for labeling. However, this way it cannot be guaranteed that we obtain representative training data. This is because for a given entity from the first dataset, most entities from the second dataset are most likely non-matches while only few will match. Hence, a random selection of entity pairs will result in an imbalanced training set where the non-matching entity pairs will heavily outnumber the matching pairs. This phenomenon is known as the class imbalance problem and has been reported as an obstacle to the generation of good classifiers by supervised machine learning algorithms [17].

A better training strategy is the so-called static-active selection of entity pairs proposed in [6]. This method compares the entities to be resolved with some state-of-the-art string similarity measure and selects only pairs that are fairly similar according to this measure, i.e. their similarity value meets a certain threshold. By asking the user to label those entity pairs a training sample with a high proportion of matching pairs can be obtained. At the same time, non-matching entity pairs selected using this method are likely to be "near-miss" negative examples that are more informative for training than randomly selected pairs most of which tend to be "easy" non-matches. The idea is easy to implement and seems a reasonable way to obtain representative training data. We thus want to further evaluate it as a strategy for training selection. In particular, we want to empirically investigate the following questions not considered in [6]:

- To what degree can we solve the class imbalance problem, i.e. provide sufficient matching and non-matching entity pairs?
- How influencing is the choice of the threshold for the similarity measure above which pairs of entities are selected for labeling? Is there a good default threshold?
- What is the influence of the training size on EM accuracy, i.e. how few training pairs would be sufficient to obtain good results?

For this study we propose and evaluate two strategies for selecting entity pairs:

- *Threshold-Random (n,m,t)*: n object pairs are randomly selected among the ones satisfying a given minimal threshold t applying a similarity measure m. This approach is very simple but may be prone to the class imbalance problem, e.g., for lower (higher) thresholds the number of non-matching (matching) pairs may dominate the training.
- *Threshold-Equal (n,m,t)*: This strategy selects an equal number (n/2) of matching and non-matching pairs from the ones satisfying a given minimal threshold t applying

a similarity measure m. This approach may require a higher manual overhead for labeling than Threshold-Random because after n selections one of the two classes (matching / non-matching) will most likely not yet have n/2 training examples. Hence, we have to select and label further pairs until n/2 examples are available for both classes.

Alternatively to these two selection methods, entity pairs could be selected using active learning methods [23], [26]. Active learning methods attempt to iteratively identify those pairs leading to maximal performance improvements when added to the training set. These pairs are then presented to the user for labeling. The system is re-trained on the training set including the newly added labeled example. We plan to incorporate selection methods based on active learning into our framework in the future and compare it with the two selection methods proposed in this paper.

4. EXPERIMENTAL EVALUATION

We present a comprehensive evaluation of the proposed training selection methods within the presented framework for real-life datasets, including bibliographic references from Google Scholar. We also investigate the impact of training size on the effectiveness of different learners for automatically constructing EM strategies.

We first outline the evaluation setting, in particular the chosen datasets, match algorithms and learners. Section 4.2 evaluates the Threshold-Random and Threshold-Equal approaches for training selection. We then comparatively evaluate the effectiveness of the learners (Section 4.3) and different match configurations (Section 4.4) for different training sizes.

4.1 Evaluation Setting

4.1.1 Datasets

We evaluate our approach for four match tasks for which the perfect match result is known. Table 1 indicates for each task the number of involved objects, the number of available attributes and the number of correspondences in the perfect match result. The two largest match tasks come from the bibliographic domain and involve subsets of three real-life data sources: Google Scholar (Scholar), DBLP and ACM Digital Library (ACM). We will provide more details on the associated match tasks below. The other (smaller) match tasks cover different domains and have been taken from the RIDDLE repository[1]. The Parks task consists of two data sources with names of parks; only one attribute is provided per source. The Restaurant problem compares restaurant listings from two restaurant guides. Both sources have four attributes: name, address, city and cuisine. We applied our framework to the remaining RIDDLE match tasks as well but observed similar results than for two selected tasks.

The bibliographic tasks are significantly larger than the RIDDLE tests and match publication sets between Scholar and DBLP (Scholar-DBLP) and between ACM and DBLP (ACM-DBLP). Scholar maintains a huge collection of publication entries automatically extracted from documents. As illustrated in the introduction this source has many data quality problems and duplicates making it very challenging to perform entity matching. DBLP and ACM focus on computer science publications and are manually maintained. Compared to Scholar they are of higher quality, especially DBLP. Since DBLP has almost no duplicates the EM result between Scholar and DBLP can also be used for determining the duplicates in Scholar. This is because all Scholar entries matching the same DBLP publication can be considered duplicates.

[1] http://www.cs.utexas.edu/users/ml/riddle/

Table 1: Overview of evaluation match tasks

match task	# entities		# attr.	# corresp.
	source1	source2		
Scholar-DBLP	64,263	2,616	4	5,347
ACM-DBLP	2,294	2,616	4	2,224
Parks	258	396	1	250
Restaurant	533	331	4	112

As shown in Table 1, our evaluation datasets cover 2,616 publications from DBLP, 2,294 publications from ACM and 64,263 publications from Google Scholar. We thus have up to 169 million entity pairs (Scholar-DBLP) in the Cartesian product between these datasets. We applied a simple blocking strategy based on a trigram similarity check with a low threshold to eliminate most pairs which are clearly non-matches. The reduced datasets contain about 607,000 (Scholar-DBLP) and 494,000 (ACM-DBLP) entity pairs. To determine the quality of the entity matching strategies we manually determined the "perfect" EM results mappings with the cardinalities shown in Table 1.

4.1.2 Match algorithms

For the evaluations we consider attribute matchers based on generic string similarity measures. Each matcher thus specifies one of the available string similarity functions as well as the pair of attributes of the input datasets for which the similarity is computed. The corresponding attributes from the input datasets have been manually determined beforehand. Table 1 indicates that between one and four attribute pairs could be used for matching (e.g., the bibliographic attributes title, authors, venue, and year). We use the following seven string similarity measures (for detailed descriptions see, e.g., [10], [13]): EditDistance, Cosine, Jaccard, Jaro-Winkler, Monge-Elkan, Trigram and TF/IDF. Applying the seven similarity functions to the available attributes gives a total of 28 attribute matchers for the bibliographic and Restaurant test cases and 7 matchers for the Parks problem.

4.1.3 Learners

We evaluate three supervised learners which have been utilized previously for entity matching, namely decision trees, logistic regression and Support Vector Machine (SVM). In addition, we evaluate a ***multiple learning approach*** combining the three single learners through voting. Two entities are considered a match if at least two of the three learners classify it as a match (majority consensus). The motivation is that the combined approach may compensate weaknesses of individual learners. For our experiments we use the learner implementations provided by RapidMiner, formerly Yale [20], a free open-source environment for machine learning algorithms.

4.1.4 *Evaluation Measures*

As in many other EM evaluations, we measure the quality of the match strategies with the standard measures precision and recall, and F-measure with respect to the manually determined "perfect" mappings. The three measures are formally defined as follows: Let M_p be the set of correspondences that an EM strategy identifies. Let M_a be the set of all correct correspondences from the set of entities to be resolved. Precision is defined as $P = |M_p \cap M_a| / |M_p|$ recall $R = |M_p \cap M_a| / |M_a|$, and F-measure = $(2P \cdot R)/(P + R)$.

4.1.5 *Manual baseline strategies*

For comparison with the automatically generated match strategies we manually configured a baseline strategy for each match task.

For the *bibliographic match tasks* we evaluated 14 configurations (thresholds 0.5 and 0.8) for the seven similarity measures and 18 configurations using the trigram similarity measure for two threshold values (0.5 and 0.8) either on one attribute (title or authors) or using two attributes. The EM strategy using trigram similarity on both title and authors with a threshold of 0.5 performed reasonably well on both EM tasks (F-measure 91.4% for the ACM-DBLP task and 82.3% for Scholar-DBLP). We therefore choose this strategy as a baseline strategy.

For each of the two simpler RIDDLE match tasks we evaluated 14 attribute configurations (thresholds 0.5 and 0.8 for each of the seven similarity measures) and use the best configuration as the baseline strategy. For the Parks problem MongeElkan with threshold 0.8 performs best (92.3% F-measure) while on the Restaurant task the best configuration is Trigram with threshold 0.8 (88.1% F-measure).

4.2 Evaluation of training selection methods

We evaluate the two methods proposed for selecting entity pairs for labeling (Section 3.1), *Threshold-Random (n,m,t)* and *Threshold-Equal (n,m,t)*. Since we want to minimize the manual labeling work as much as possible we focus on the selection of only few training pairs (n = 20, 50, 100, and 500). The threshold t is varied from 0.4 to 0.8.

Fig. 4 displays the F-measure results for the four match tasks Scholar-DBLP, ACM-DBLP, Parks and Restaurant. The results for *Threshold-Random* (*Threshold-Equal*) are shown in the top (lower) four diagrams. The results are achieved with Trigram attribute matchers and SVM as the learner. Other match configurations and learners gave similar relative results. For comparison the F-measure results for the manually determined baseline match configurations are also shown.

We first observe that despite the use of very small training sets the constructed EM strategies outperform the baseline strategy in many cases. The simple *Threshold-Random* approach is somewhat more dependent on the training size n and the choice of the threshold than *Threshold-Equal*, especially for the ACM-DBLP and Restaurant problems. As indicated in the top diagrams of Fig. 4, the achieved F-measure results and thus the training quality drop sharply for threshold values of 0.7 or higher for the ACM-DBLP and the Restaurant problems. This is because the ACM-DBLP and Restaurant problems all involve relatively clean datasets for matching, thus high threshold values mainly select matching entity pairs. Hence *Threshold-Random* runs into a class imbalance problem with many matching and few non-matching pairs. Additionally, the non-matching entity pairs selected with a high threshold may be rare outliers and we risk that the learner is

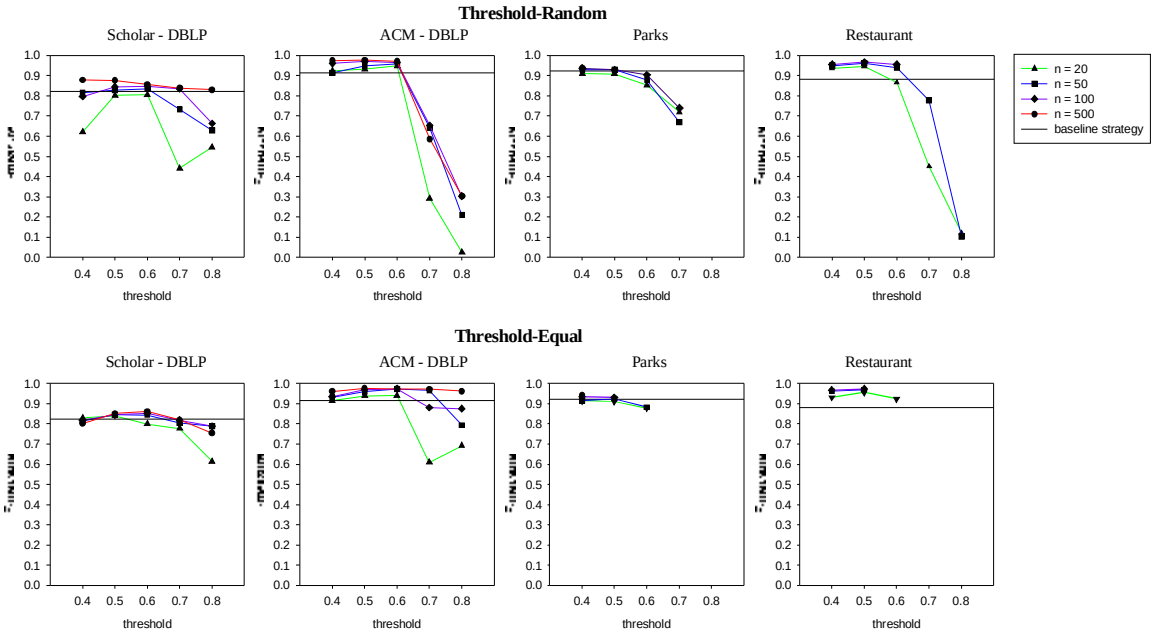


Figure 4: Comparison of Threshold-Random and Threshold-Equal training selection

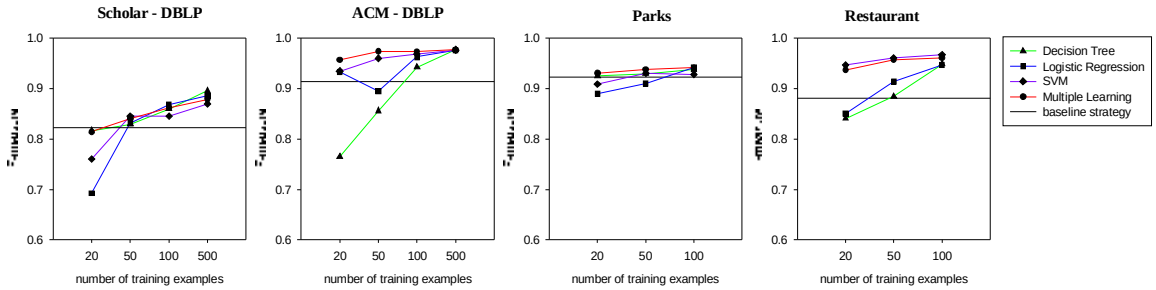


Figure 5: Comparison of different learners

overfitted to those special cases preventing to classify other entity pairs correctly. For threshold values 0.4 to 0.6, *Threshold-Random* achieves very good results for ACM-DBLP and Restaurant. For Scholar-DBLP the training size n seems more significant than the choice of threshold, but the best results are again achieved for t between 0.4 and 0.6.

For all four match tasks, *Threshold-Equal* also performs best in the threshold range 0.4 to 0.6. However, it is much more stable than *Threshold-Random* with respect to different training sizes and threshold values because it avoids the class imbalance problem. This comes at the prize of a higher manual labeling effort to guarantee equal numbers of matching and non-matching entity pairs in the resulting training set. For example, for obtaining 20 entity pairs with equal numbers of matching and non-matching pairs for the Scholar-DBLP match problem using a threshold of 0.4 we had to label 400 pairs, that is a 20 times higher effort than with *Threshold-Random*. For the two smaller problems, Parks and Restaurant, we could not even find the required number of non-matches for Threshold-Equal and threshold values of 0.7 or higher, indicating a limited applicability of this method for small match tasks.

We also observe that *Threshold-Equal* and *Threshold-Random* achieve similar maximal F-measure values of about 86% for Scholar-DBLP, 96% for ACM-DBLP, 93% for Parks and 97% for Restaurant (n=500). We therefore propose the cheap and more general *Threshold-Random* approach with a threshold of 0.5 as the default method for training selection and use it in our further evaluation. We repeated the experiments for other similarity measures than Trigram to select training samples. We found out that TF/IDF performs similarly well and exhibits a comparable behavior w.r.t. the training sizes and thresholds. Hence this measure would be an alternative choice for use with Threshold-Random.

4.3 Learner evaluation

Next, we want to evaluate the influence of the training size on the relative quality of the four supervised learners: decision tree, logistic regression, SVM and the multiple learning approach. We used the same match configurations as for the evaluation of the training selection methods in the previous section.

Fig. 5 shows the F-measure results for the four match tasks achieved with the automatically generated EM strategies and different training sizes (x-axis). Training sizes vary between 20 and 500 entity pairs. For the two smaller match problems the near-optimal results are already achieved for 100 or fewer pairs. We observe that all learners benefit from increasing the training size especially for the more difficult Scholar-DBLP problem. For this task choosing only 20 training entities is clearly insufficient but all four learners match or exceed the baseline performance already for 50 training pairs. For the ACM-DBLP and Restaurant tasks only 20 training pairs were sufficient for SVM and the multiple learner approach to outperform the manually determined baseline approach. The Parks task offered little optimization potential since its data sources provide only one attribute. Furthermore, the chosen baseline configuration performed already very well and could not be much improved by the consideration of additional matchers on the attribute.

Altogether, we see that the multiple learner approach is especially stable and among the top-performing approach for all four match tasks. This shows that it is able to combine the advantages of the different approaches. From the remaining learners, SVM performed best in most cases.

Compared to the multiple learning and SVM approaches the decision tree and logistic regression learners perform worse and rather unstable. On the ACM-DBLP and the Restaurant match tasks they both need a higher number of training examples to construct an effective EM strategy. For less than 100 training pairs the decision tree learner is clearly inferior to the other three learners and even the baseline strategy on the ACM-DBLP match task. The decision tree learner is among the best approaches for two tasks (Scholar-DBLP, Parks) but performs worst for the two other problems.

We therefore conclude that for a small amount of training data the multiple learning approach or SVM can already be very effective and outperform (many) manually configured EM strategies. Furthermore, they are to be preferred to using the decision tree or logistic regression learners.

4.4 Influence of match configuration

To examine the usefulness of using a greater selection of matchers we now compare the performance of the four learners for two match configurations. For this comparison we focus on the more challenging match task Scholar-DBLP. In addition to the previously considered configurations with the four trigram matchers we now consider all (7) similarity measures resulting in a total of 28 matchers for the four attributes.

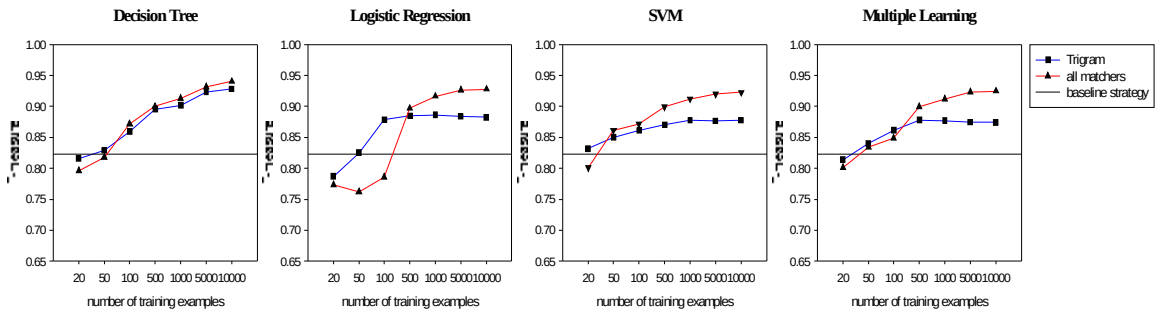


Figure 6: Evaluation of different matcher selections

Fig. 6 contains a separate diagram for each of the four learners comparing the F-measure results for the two match configurations and different training sizes. The curves indicate that for small amounts of training examples the increased choice of matchers does not significantly improve performance but mostly decreases performance. This indicates that the learners need more training to effectively deal with the much increased solution space for selecting, ordering and weighting the applicable matchers. We therefore studied additional training sizes of up to 10,000 training pairs for this experiment. With more training, all learners eventually benefit from the increased number of matchers and outperform the EM strategies based on only four matchers. The improvements are especially significant for the three learners SVM, logistic regression and multiple learners for 500 and more training samples. SVM achieves this already for 50 training examples. The decision tree learner exhibits similar performance for both match configurations so that it cannot effectively utilize the additional matchers. On the other hand, it benefits the most from increased training sizes, especially for the trigram matcher, and achieves the absolute best F-measure (94.1% for n=10,000).

To summarize we observe that our tuning framework can utilize a large choice of matchers to improve performance albeit at the expense of more training. SVM needs the least training data to utilize the increased optimization potential. For large training sizes decision tree performed best.

4.5 Concluding remarks

Our findings so far suggest several default configurations and alternatives for our framework to generate promising EM strategies for a new match problem. For training selection, the threshold-random method should be used to generate n training samples. A good default seems n=100; fewer training samples may suffice for small-sized match problems or comparatively clean data sources (e.g., ACM-DBLP, Restaurant). The matcher configuration may just apply all available matchers (default) or a restricted set of matchers, e.g., with only a subset of string similarity measures such as Trigram and TFIDF. As we have seen, restricting the number of matchers reduces the dependency on larger training sizes; it will also reduce the computational overhead compared to the execution of all matchers. For the learners, both SVM and multiple learning could be applied to generate EM strategies. In future work we will study the tradeoffs between match accuracy and execution time and how suitable heuristics can consider these tradeoffs in choosing from several EM strategies. We will also study methods to automatically propose suitable match configurations, e.g., by using the training samples to determine ineffective matchers beforehand and eliminate them for learning EM strategies

5. Related work

There has been a large body of research on entity matching and its variations. For a recent survey and tutorial see [13] and [18], respectively. Most previous studies used a single match approach like threshold-based attribute matching (similarity join [7]), clustering [19], or context-based matching [1], [11], [29].

A single match approach typically performs very differently for different domains and match problems. For example, it has been shown that there is no universally best string similarity measure [16], [24]. Hence, one needs to customize individual matchers (e.g., selection of attributes and similarity functions) and/or combine several matchers for improved accuracy. Several toolboxes and frameworks support such customization to specify an entity matching strategy, e.g., by a workflow or operator tree of several matchers [14], [12], [25], [27]. However, these specifications typically have to be performed manually which is a very difficult and time-consuming administration task even for tuning experts.

The main approach to automating the generation of entity matching strategies is the use of supervised (training-based) approaches or learners. They aim at adapting the matcher configuration and combination to specific match problems thereby supporting tuning. The learners supported in our framework have already been investigated for entity matching in several previous studies, namely decision trees [15], [23], [26], [28], logistic regression [22], and Support Vector Machine [5], [21], [23]. In addition, maximum entropy [9], and Naïve Bayes [23] approaches have been considered.

Bilenko and Mooney have shown in an empirical evaluation [6] that the Support Vector Machine is more accurate than a decision tree learner. They used SVM at two levels. At the first level they tune four matchers, namely edit distance applied to four attributes of the evaluation dataset. At the second level they apply the Support Vector Machine on the tuned matchers from the previous step.

While the previous studies showed already the potential of supervised methods for tuning, no comprehensive framework such as ours has been studied so far. Our framework allows us to utilize multiple training selection methods, matcher

configurations and learners and can easily add additional methods for a comparative evaluation, e.g., like the proposed multiple learner approach. As already discussed in Section 3, most previous evaluations have provided only limited information on how training examples where acquired and how many were necessary to achieve the stated results making it difficult to judge whether good results were due to the approach or clever (time-consuming) manual training data selection.

Recently, Chaudhuri et. al. [8] presented a fast training-based tuning approach for the combined use of several matchers. The idea is to construct (match) operator trees, corresponding to the Or-connection of several And-connected similarity predicates on attributes. These predicates are automatically determined by a recursive divide and conquer strategy on the labeled training data. An implementation based on SQL-Server2005 achieved significantly better execution times than for a standard SVM implementation at a comparable accuracy. The authors do not discuss how they selected the training data; the number of training examples varied for different datasets. The study is complementary to ours since their approach could be plugged into our framework as a promising learner to generate EM strategies.

Most previous work focuses on maximizing the quality of entity matching, in particular precision and recall. Supporting fast entity matching is similarly important especially for large match problems. Several approaches have been developed for blocking (see [3] for an overview) to reduce the search space for entity matching from the Cartesian product to a small subset of the most likely correspondences. Such methods can be plugged into our framework. Similarly we can incorporate proposed enhancements to speed-up individual matchers and learners, e.g., by utilizing set similarity join techniques [2] or the optimizations proposed in [4].

6. SUMMARY AND OUTLOOK

We proposed and evaluated two generic methods for automatically selecting training data to be labeled. The training selection methods are part of a powerful generic framework called STEM that supports the automatic construction of entity matching strategies. The framework addresses the problem of how to automatically configure and combine several matchers within an entity matching strategy.

Our evaluation on diverse datasets showed that automatically constructed EM strategies using the proposed training selection methods can significantly outperform manually configured combined strategies. This is especially true for difficult match tasks and often possible for small training sizes incurring a low labeling effort. The evaluation of several learners revealed that the SVM and the multiple learning approach produce the most stable results even for small training sizes. By contrast decision trees perform well only for large amounts of training data.

In future work, we will further investigate the new framework and focus on both EM quality and runtime performance.

7. REFERENCES

[1] Ananthakrishna, R., Chaudhuri, S., and Ganti, V.: Eliminating Fuzzy Duplicates in Data Warehouses. In *Proc. of VLDB*, 2002.

[2] Arasu, A., Ganti, V. and Kaushik, R..: Efficient exact set-similarity joins. In *Proc. of VLDB*, 2006.

[3] Baxter, R., Christen, P, and Churches, T.: A comparison of fast blocking methods for record linkage. In *Proc. of ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[4] Benjelloun, O, Gracia-Molina H., Kawai, H., Larson, T. E., Minestrina, D., Su, Q., Thavisomboon, S., and Widom, J.: Generic Entity Resolution in the SERF project. *IEEE Data Engineering Bulletin* , Vol. 29, Number 2, 2006.

[5] Bilenko, M. and Mooney, R. J.: Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD*, 2003.

[6] Bilenko, M. and Mooney, R. J.: On Evaluation and Training-Set Construction for Duplicate Detection. In *Proc. of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003..

[7] Chaudhuri, D., Ganti, V., and Kaushik, R.: A Primitive Operator for Similarity Joins in Data Cleaning. In *Proc. of. ICDE*, 2006.

[8] Chaudhuri, S., Chen, B.-C., Ganti,V., and Kaushik, R.: Example-driven design of efficient record machting queries. In *Proc. of VLDB*, 2007.

[9] Cohen, W. W. and Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In *Proc. of ACM SIGKDD*, 2002.

[10] Cohen, W. W., Ravikumar, P., and Fienberg, S. E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proc. of Workshop on Information Integration on the Web (IIWeb)*, 2003.

[11] Dong, X., Halevy, A., and Madhavan, J.: Reference reconciliation in complex information spaces. In *Proc. of ACM SIGMOD*, 2005.

[12] Elfeky, M. G., Elmagarmid, A.K., and Verykios, V.S.: TAILOR: A Record Linkage Tool Box. In *Proc. of ICDE*, 2002.

[13] Elmagarmid, A.K., Ipeirotis, P.G., and Verykios, V.S.: Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19(1), 2007.

[14] Galhardas, H., Florescu, D., Shash, D., and Simon, E.: AJAX: An Extensible Data Cleaning Tool. In *Proc. of ACM SIGMOD*, 2000.

[15] Ganesh, M., Srivastava, J., and Richardson, T.: Mining entity-identification rules for database integration. In *Proc. of SIGKDD*, 1996.

[16] Guha, S., Koudas, N., Marathe, A., and Srivastava, D.: Merging the results of approximate match operations. In *Proc. of VLDB*, 2004.

[17] Japkowicz, N. and Stephen, S.: The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis Journal* 6(5), 2002.

[18] Koudas, N., Sarawagi, S., and Srivastava, D.: Record linkage: Similarity measures and algorithms. In *Proc. of ACM SIGMOD*, 2006.

[19] McCallum, A., Nigam, K., and Ungar, L. H.: Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proc. of ACM SIGKDD*, 2000.

[20] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M, and Euler, T.: Yale: Rapid Prototyping for Complex Data Mining Tasks. In *Proc. of ACM SIGKDD*, 2006.

[21] Minton, S. N, Nanjo, C., Knobloch, C. A., Michalowski, M., and Michelson, M.: A Heterogeneous Field Matching Method for Record Linkage. In *Proc. IEEE International Conference on Data Mining*, 2005.

[22] Pinheiro, J. C., and Sun, D. X.: Methods for linking and mining massive heterogeneous datasets. In *Proc. of ACM SIGKDD*, 1998.

[23] Sarawagi, S., and Bhamidipaty, A.: Interactive deduplication using active learning. In *Proc. of ACM SIGKDD*, 2002.

[24] Sarawagi, S. and Kirpal, A.: Efficient set joins on similarity predicates. In *Proc. of ACM SIGMOD*, 2004.

[25] Shen, W., DeRose, R., Vu, L., Doan, A., and Ramakrishnan, R.: Source-aware Entity Matching: A Compositional Approach. In *Proc. of ICDE*, 2007.

[26] Tejada, S., Knoblock, C. A., and Minton, S.: Learning object identification rules for information integration. *Information Systems Journal*, 26(8), 2001.

[27] Thor, A., and Rahm, E.: MOMA - A Mapping-based Object Matching System. In *Proc. of CIDR*, 2007.

[28] Verykios, V. S., Moustakides, G.V., and Elfeky, M. G.: A Bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12(1), 2003.

[29] Weis, N., and Naumann, F.: DogmatiX tracks down Duplicated in XML. In *Proc. of ACM SIGMOD*, 2005.

Record Linkage as DNA Sequence Alignment Problem

Yoojin Hong
Penn State Univ., USA
yuh108@psu.edu

Tao Yang
Penn State Univ., USA
tuy104@psu.edu

Jaewoo Kang*
Korea Univ., Korea
kangj@korea.ac.kr

Dongwon Lee†
Penn State Univ., USA
dongwon@psu.edu

ABSTRACT

Since modern database applications increasingly need to deal with dirty data due to a variety of reasons (e.g., data entry errors, heterogeneous formats, and ambiguous terms), numerous research efforts have focused on the **(record) linkage** problem to determine if two entities represented as relational records are approximately the same or not. In this paper, we propose a novel idea of using the popular gene sequence alignment algorithm in Biology – BLAST. Our proposal, termed as the BLASTed linkage, is based on the observations that: (1) both problems are variants of approximate pattern matching, (2) BLAST provides the statistical guarantee of search results in a scalable manner – a greatly lacking feature in many linkage solutions, and (3) by transforming the record linkage problem into the gene sequence alignment problem, one can leverage on a wealth of advanced algorithms, implementations, and tools that have been actively developed for BLAST during the last decade. In translating English alphabets to DNA sequences of `A`, `C`, `G`, and `T`, we study four variations: (1) *default* – each alphabet is mapped to nucleotides under 1, 2, and 4-bit coding schemes, (2) *weighted* – tokens are elongated or shortened proportional to their importance, making important tokens longer in the resultant DNA sequences, (3) *hybrid* – each token's lexical meaning as well as its importance are considered at the same time during translation, and (4) *multi-bit* – tokens are selected for any of 1, 2, and 4-bit coding schemes based on the cumulative distribution functions of their importance. The efficacy of our proposed BLASTed linkage schemes are experimentally validated using both real and synthetic data sets.

1. INTRODUCTION

Poor quality data is prevalent in databases due to a variety of reasons, including transcription errors, data entry mistakes, lack of standards for recording databases, etc. To fix such errors, considerable recent work has focused on the **(record) linkage problem**[1], i.e., determine whether or not two entities represented as relational records are approximately the same. The linkage problem frequently occurs in data applications (e.g., digital libraries, search engines, customer relationship management) and gets exacerbated when data are integrated from heterogeneous sources. For instance, a customer address table in a data warehouse may contain multiple address records that are all from the same residence, and thus need to be consolidated. For another example, imagine integrating two digital libraries, say DBLP and CiteSeer. Since citations in two systems tend to have different formats, identifying all matching pairs is not always straightforward.

The linkage problem has been extensively studied in various disciplines. By and large, contemporary approaches have mostly focused on how to identify similar records "accurately" by designing new methods or models in a particular situation. However, often, newly-developed linkage solutions are hard to apply to data sets of different scenarios, let alone using additional tools to visualize or analyze the results further. One way to approach the problem is to develop a suite of linkage algorithms and tools for generic usage so that the developed linkage solutions can be used in a variety of situations. Another way is to extend an existing generic solution to solve the linkage problem so that one can leverage on the development of the generic solution and its user base. In this paper, we take the latter approach and aim at solving the linkage problem by applying one of such popular existing solutions drawn from Bioinformatics, called BLAST (*B*asic *L*ocal *A*lignment *S*earch *T*ool) [1].

The BLAST, developed in 1990, is one of the most popular (and best cited) algorithms for aligning biological sequence information such as nucleotides of DNA sequences and amino acid sequences of proteins. In its essence, given a query sequence q, BLAST finds top-k most similar sequences above a threshold from a database of sequences D using approximation[2] heuristics based on Smith-Waterman algorithm [20]. Our decision to use BLAST for the linkage problem is based on the observations that: (1) Both problems are variants of approximate pattern matching. That is, comparing two address records of "John Doe, 110 E. Foster

*Partially supported by Korea University grant.
†Partially supported by IBM and Microsoft gifts.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

[1]The record linkage problem is also known as other names such as *merge-purge* [9], *de-duplication* [18], or *entity resolution* [19] problem. Throughout the paper, to be consistent, we will use the *linkage problem* to refer to all these problems.
[2]It is reported that BLAST is able to produce results that are slightly less accurate than what Smith-Waterman algorithm [20] would generate but over 50 times faster [1].

Notation	Description
q_r, q_s	a query record and sequence
d_r, d_s	a duplicate record and sequence
D_r, D_s	a database of records and sequences
t_1, t_2 ..	tokens
tf	term frequency
tf/idf	term freq./inverse document freq.

Table 1: Notations used.

Ave. State College, PA, USA" and "Dr. J. E. Doe, 110 East Foster Franklin Avenue, University Park, PA 16802" in the image are likely related to one another because an alignment between the sequences `ATATCG` and `ATTAGC` using BLAST; (2) The alignment results from BLAST provide a robust similarity measure of *S-score* as well as a sound reliability measure of *E-value*[3]. Such a statistical guarantee has been greatly lacking in existing linkage solutions. Furthermore, BLAST is known for its fast running time and ability to handle a large amount of sequence data; and (3) BLAST has a wealth of advanced algorithms (e.g., nucleotide-nucleotide, protein-protein, and section-specific versions), implementations (e.g., NCBI BLAST, FPGA-based BioBoost BLAST, and open source version), and tools (e.g., KoriBLAST for visualization and Parallel BLAST for parallel processing [17]) to leverage on. Therefore, if one can successfully transform the linkage problem to the gene sequence alignment problem, one can have an immediate access to a vast number of application software to use.

In the investigation of BLAST for the linkage problem, in particular, we aim to have the following *desiderata* to maximize the impact of study: (1) The kernel of BLAST algorithm and implementation should not be changed to make existing tools remained as useful. Instead, our proposal sits atop BLAST algorithm and manipulates query and database sequences; and (2) The proposed scheme should be able to identify matching records fast as well as accurately. That is, both accuracy and speed of the solution should be the evaluation metrics.

Our contributions are as follows:

- To the best of our knowledge, this is the first effort to solve the linkage problem using the BLAST technology. We formally introduce the linkage problem and propose a framework to show how to apply BLAST algorithm.
- We propose and evaluate four different BLAST based linkage solutions to translate string data into DNA sequences while considering the relative importance of tokens in the string data.
- We compare the proposed BLAST based schemes against popular approximate pattern matching schemes such as *Jaro*, *Jaccard*, and *SoftTFIDF* and report promising results of our proposal in both speed and accuracy.

[3]E-value is the probability indicating that the S score of the current sequence could be due to by chance using the given database. For instance, an E-value less than e^{-5} of an alignment means that the alignment is very unlikely to occur by chance – thus a good match.

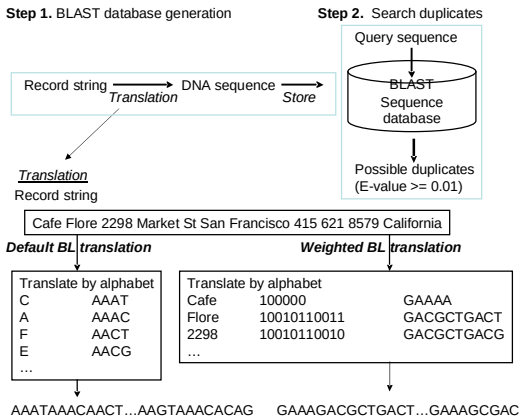


Figure 1: Overview of the BLASTed Linkage.

2. OVERVIEW OF PROBLEM AND SOLUTION

Throughout the paper, we use notations in Table 1.

Problem Overview. The linkage problem can be modeled as a *selection* problem (i.e., select top-k similar records) as well as a *threshold* problem (i.e., find all similar records above a threshold). For illustration purposes, we briefly describe the selection version of the linkage problem as follows. Imagine that given a query record q_r, there are k true duplicates, d_1, ..., d_k, in the database of records D_r. Then, the goal of record linkage is to obtain the true duplicates from the retrieved top-k records. In the best scenario, if all k true duplicates are retrieved, then we get the precision and recall of 1.0. Note that in this paper, we adopt both the selection approach and the threshold approach for further experimental analysis.

Given a set of query records Q_r and a database of records D_r, a naive nested-loop style algorithm with quadratic running time to find all duplicate records is:

for each record $r \in Q_r$
  for each record $s_i \in D_r$
    compute $dist(r, s_i)$;
  return $\{s_1, ..., s_k\}$ with the lowest $dist(r, s_i)$;

Note that what we study in this paper is *not* the algorithmic improvement of the linkage problem *nor* the study of $dist()$ function therein. Therefore, such an optimization or modeling issue is not further discussed in the paper.

Solution Overview. The basic framework of our approach, termed as the "BLASTed linkage" or "BL" in short, is illustrated in Figure 1. All strings in database D are first translated to DNA sequences by converting each English alphabet character to pre-assigned one, two, or four nucleotides under 1, 2, or 4-bit coding scheme, respectively. For instance, the alphabet "A" is re-written to nucleotides of `T`, `GT`, or `AAAC` under 1, 2, or 4-bit coding scheme, respectively. Then, the query sequence q is also translated in a similar manner. Finally, BLAST is invoked to find similar sequences to q from D. In addition to this default BLASTed linkage technique, we also propose three other variations depending on how we "translate" strings to DNA sequences –

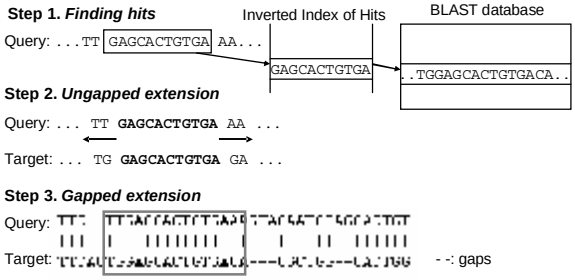


Figure 2: Three steps of BLAST.

Letter vs. Nucleotides (4-bit)							
A	AAAC	B	AAAG	C	AAAT	D	AACC
E	AACG	F	AACT	G	AAGC	H	AAGG
I	AAGT	J	AATC	K	AATG	L	AATT
M	ACAC	N	ACAG	O	ACAT	P	...

Letter vs. Nucleotides (1-bit/2-bit)							
A	T/GT	B	A/AG	C	C/CA	D	T/GC
E	A/AA	F	C/CC	G	T/GG	H	G/CT
I	G/GA	J	C/CG	K	A/AG	L	C/AT
M	T/GT	N	A/AA	O	T/GG	P	...

Table 2: Conversion tables. (Partial)

weighted, *hybrid*, and *multi-bit* BLASTed linkage.

- In the *default BLASTed linkage* (in Section 3.1), each alphabet character in record strings is translated to pre-assigned number of nucleotides.
- In the *weighted BLASTed linkage* (in Section 3.2), each token in record strings is either elongated or shortened, proportional to their associated weights, to make important tokens longer in DNA sequences.
- In the *hybrid BLASTed linkage* (in Section 3.3), each token's lexical meaning is incorporated into the translation in addition to the importance such as their weights or ranks.
- In the *multi-bit BLASTed linkage* (in Section 3.4), each token is assigned to any of 1, 2 or 4-bit coding schemes based on the cumulative distribution functions of their importance.

3. MAIN PROPOSAL: BLASTed Linkage

The original BLAST algorithm performs three basic steps to align similar sequences. These three steps are illustrated in the format of DNA sequences in Figure 2. Given a query sequence q and a database of sequences D, consider the candidate sequence c ($c \in D$). Then,

1. ***Exact Match***: BLAST first searches all sequences from D that share "hits" with q, where hits are the exactly matching continuous letters of nucleotides between c and q. By default, in BLAST, the size of a hit for DNA sequence is 11 nucleotides.

2. ***Ungapped Extension***: In the second step, if c has at least two hits in close distance, BLAST extends the two hits without gaps. During the extension, BLAST increases the score for matching pairs and decreases it for mismatching pairs. If the score drops below a certain threshold, then the process stops and the final result becomes the High-scoring Segment Pairs (HSP). The box in step 3 of Figure 2 represents HSP from the ungapped extension step.

3. ***Gapped Extension***: Last, BLAST extends the HSP (with matching score above the threshold) with gaps as long as the final score does not drop below a threshold, which is larger than the threshold of ungapped extension.

At each step, BLAST can eliminate some candidate sequences with a final alignment score below the thresholds to reduce the cost of expensive dynamic programming. That is, the thresholds for ungapped and gapped extension help BLAST avoid comparing the entire sequence.

The key contribution of the BLASTed linkage is to enable BLAST to do the "linkage" of query and database records. To enable this, one needs to translate query and database records into DNA sequences of `A`, `C`, `G`, and `T`. For this translation, we study four variations below.

3.1 Default BLASTed Linkage

In the default BLASTed linkage, each alphabet letter in record strings is translated to pre-assigned number of one, two, or four nucleotides depending on 1, 2, or 4-bit coding schemes, respectively. In this paper, we used three different coding schemes as follows:

4-bit scheme: An alphabet letter is translated to four nucleotides, as shown in Table 2 (top). Note that the assignment of four nucleotides to an alphabet is done such that the assigned nucleotides become "prefix-free". A prefix-free code such as Huffman code is a code where no code word is a prefix of any other codes. For instance, a code set {0, 1, 01} is not prefix free since "0" is the prefix of "01", but a code set {000, 010, 111} is. We use 4-bit coding of Table 2 (top) borrowed from [13]. However, note that any other prefix-free assignment would work correctly. Using 4-bit encoding, note also that after the translation, DNA sequences are always four times longer than original English strings. In the translation, ignoring upper/lower cases, we consider 37 (= 26 + 10 + 1) cases – i.e., 26 cases for 26 English alphabets, 10 cases for 10 numbers (e.g., 0 to 9), and one case for all special characters (e.g., `@`, `#`, `$`). we used the 4-bit scheme, by default.

1-bit/2-bit schemes: Our eventual goal in this translation is *not* to have good representation of nucleotides, rather to translate two *similar* (*dissimilar*) records to two *similar* (*dissimilar*) DNA sequences so that both can be aligned correctly by BLAST. That is, whether "*F*atabase" is translated to `AACTAAACACGGAAACAAAGAAACACGCAACG` in 4-bit coding or `CTTTATGA` in 1-bit coding is not important as long as both DNA sequences can be aligned well with DNA sequences for "*D*atabase" if we use BLAST to distinguish them. Based on this observation, we also propose to use one or two nucleotide(s) to represent each alphabet letter. For example, we can group 37 cases into 4 groups, and assign one of `A`, `C`, `G`, and `T` to each group. Similarly, we can also group 37 cases into 16 groups to use two nucleotides per group. This coding scheme is shown in Table 2 (bottom).

The actual assignment of 1-bit or 2-bit nucleotides letters to each group is done randomly since no a priori information is introduced in the experimentation. However, one can exploit further heuristic features in the assignment. For in-

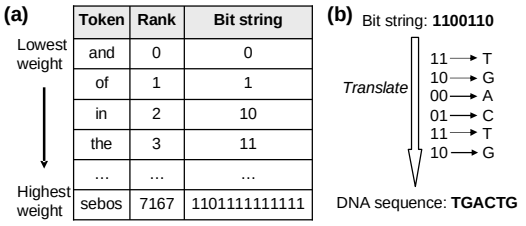


Figure 3: The sliding window coding scheme in the weighted BLASTed linkage: (a) tokens are sorted by their weights, (b) bit string is translated to DNA sequence by the sliding window of size 2.

stance, alphabet letters in the neighborhood on keyboards can be assigned to the same group. Finally, note that compared to the 4-bit coding scheme which is a lossless scheme, 1-bit and 2-bit schemes are lossy – i.e., we are unable to translate translated strings back to the original ones. Despite their lossy translation, we will show that 1-bit and 2-bit coding schemes can achieve acceptable accuracy with noticeable speed-up.

3.2 Weighted BLASTed Linkage

By the coding schemes of default BLASTed linkage, longer English letters will always have longer DNA sequences, regardless of bit schemes. For instance, when an article title "Improving RNA ..." is translated, the word "Improving" will have 36 nucleotides `AAGTACACACCCACCTACATACTCAAGTACA GAAGC` while the second word "RNA" has 12 nucleotides `ACCTACAGAAAC` under 4-bit scheme. Therefore, if there is another title such as "Improving Stem Cell ...", then BLAST is likely to find a similar match since 33 nucleotides between two sequences (corresponding to "Improving") are identical. However, in both titles, the other words such as "RNA" or "Stem Cell" carry more important meaning and a match is most likely less intuitive. For titles with titles having either "RNA" or "Stem Cell", not "Improving."

To capture this intuition, in the weighted BLASTed linkage, each word token t of a record string s from a database of sequences D is associated with an importance weight w using the *tf-idf* weight (term frequency-inverse document frequency) of the traditional weighting scheme in Information Retrieval. This tf-idf weight is a statistical measure to estimate how important a word token t in the string s is within a database D such that it increases proportionally to the number of times t occurs in s and is offset by its frequency in D. For instance, for a record string "Improving RNA ...", the token "Improving" would have a lower tf-idf weight than what the token "RNA" would have, assuming that the token "Improving" occurs frequently among entire corpus of D.

Given tf-idf weight, each token can be converted to corresponding bits using the *sliding window coding*, as illustrated in Figure 3. The details are as follows:

1. Word tokens are sorted in ascending order according to their tf-idf weights – from the *least* important to the *most* important token. For a token t with the rank r (starting from 0) in this order, then, a binary string corresponding for the rank r is used as the encoded string for the token t. For instance, the third lowest rank word "in" in Figure 3(a) has the rank of #2 (since the rank starts from 0), and is thus converted to its corresponding binary representation of `11` (i.e., `11` = "third lowest"). In addition to this simple ranking-based encoding method, we also tried more sophisticated method using equi-distance histograms. For instance, using 1,000 equi-distance bins (i.e., bin #1 for weights in [0.000, 0.001), bin #2 for weights in [0.001, 0.002), etc.), all tokens are assigned to appropriate bins with similar weights (thus similar importance), and subsequently all tokens in the same bin are encoded into the same binary string (i.e., `0` for bin #0, `1` for bin #1, and `10` for bin #2, etc.)[4].

2. Once a binary bit string representation for the token t is ready, a sliding window w slides down the bit string, one bit each time, and encodes bits within the window to corresponding nucleotides. When w =2, the sliding window can have four kinds of bit strings, {`00`, `01`, `10`, `11`} (ignoring `0` and `1`), which can be encoded to `A`, `C`, `G`, and `T`, respectively. As shown in Figure 3(b), based on the binary bit string `1100110` for a particular token, six 2-bits of {`11`, `10`, `00`, `01`, `11`, `10`} are generated in the sliding window, and six letter nucleotides `TGACTG` become the final DNA sequence.

Note that since more important tokens with higher tf-idf weights are encoded to longer bits, their final DNA sequences will be much longer as well. Therefore, when BLAST aligns sequences, such elongated DNA sequences will play a more significant role than they would in the default BLASTed linkage.

One potential caveat of the rank based weighted BLASTed linkage is that each unique token in the corpus D will have unique DNA sequences assigned at the end. For instance, when there are two tokens t_1 ("Database") and t_2 ("Fatabase") in D, since they may have different tf-idf weights, their respective ranks may be extremely different. In turn, this will yield two completely different final DNA sequences although the two tokens appear to be similar. To mitigate this problem, we can first group tokens which must be considered as the same or similar word and assign the same DNA sequence to each token group. In the experimentation, we applied the popular and Winkler method to find obvious typos first.

3.3 Hybrid BLASTed Linkage

Compared to those in the default BLASTed linkage, the final DNA sequences of tokens in the weighted BLASTed linkage are completely determined by their importance weights. To some extent, we become lost the lexical information of tokens. In other words, there might exist such situation that two tokens are completely different, but they happen to carry equal or close importance weights in the database. For instance, let us consider a citation record such as "...proceedings of VLDB ..." and the other citation record such as "...proceedings of SIGMOD ...". The words "VLDB" and the "SIGMOD " may have close importance weights in the database. Suppose the binary bit string representations for "VLDB" and "SIGMOD" are `11000` and `11001`, respectively. When they are translated using the *sliding window coding*, the final DNA sequences for "VLDB" and "SIGMOD" become `TGAA` and `TGAC`, respectively. Obviously, this may result in a similar match between these two tokens although they are completely different.

[4]In the experimentation, the histogram based scheme did not show significant improvement over the simple rank-based method, and thus is not discussed any further.

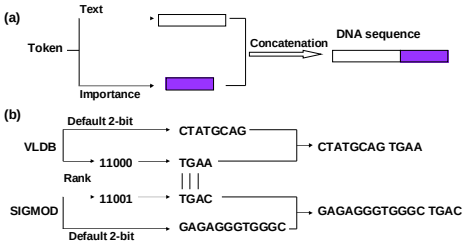


Figure 4: The coding scheme in the hybrid BLASTed linkage: (a) importance weights are appended to the lexical information of tokens, (b) comparison of token "VLDB" and "SIGMOD" using 2-bit as default coding and ranks as importance weights in the hybrid BLASTed linkage.

Motivated by this intuition, in the hybrid BLASTed linkage, we propose to incorporate the lexical information into the translation in addition to the importance weights of tokens. For each token, we append the gene sequence generated from the importance weight to the end of the gene sequence generated from the default coding scheme, as illustrated in Figure 4(a). For the above example, using this hybrid coding scheme with 2-bit as default coding and ranks as importance weights to translate DNA sequences, "VLDB" and "SIGMOD" become `CTATGCAGTGAA` and `GAGAGGGTGGGCTGAC`, respectively. (Figure 4(b)). Accordingly, the probability that BLAST finds these two tokens as a similar match is greatly reduced.

The proposed hybrid BLASTed linkage also have some advantage over the default BLASTed linkage. According to the default coding, a longer token has a longer DNA sequence based on the lexical formation. If it is more important in the corpus, then its DNA sequence generated from the importance weight is also longer. Therefore in the hybrid BLASTed linkage, by combining the two DNA subsequences, it would have even longer final DNA sequence for BLAST to distinguish. However, if it is not important in the corpus, then its DNA sequence based on the importance weight is more shorter, which in turn offsets the length of the DNA sequence generated from the default coding. Since most English words have limited length but the rank of a word in a very large database can be significantly high, the hybrid BLASTed linkage can improve the precision of pattern matching in BLAST without loss of lexical meaning of English words.

In addition, the hybrid BLASTed linkage can mitigate the synonym or typo problem which is mentioned at the end of Section 3.2. Although the two tokens "*D*atabase" and "*F*atabase" have different tf-idf weights, they can have different gene sequences based on their ranks. Their lexical formations are quite similar. Therefore, the gene sequences generated from the default coding can offset the difference between their ranks. Overall, the final DNA sequences can still be similar to reduce the chance for BLAST to find the mis-match.

3.4 Multi-bit BLASTed Linkage

In the weighted BLASTed linkage, there exists such situation that two tokens have different tf-idf weights but they have close ranks which in turn result in similar final DNA sequences after translation. The hybrid BLASTed linkage, to some extent, can mitigate this problem. However, the combination of two component gene sequences may quickly increase the length of the final DNA sequence for each token. Thus, we propose a multi-bit BLASTed linkage. The main idea is to divide all the tokens into three groups based on the actual values of tf-idf weights and then assign 1, 2 and 4-bit coding schemes to the three groups respectively to ensure that more important tokens can be translated to longer DNA sequences.

In this scheme, we regard different tokens in the corpus as possible values of a discrete random variable X, i.e., X can take on possible values $\{x_1, x_2, ..., x_n\}$ where n is total # of tokens in the database. The weight of each token is regarded as the probability of X at that particular value. Note that if the weights are not normalized, i.e., the sum of the weights are not equal to 1, we need to normalize these weights first (e.g., by dividing each weight by the total sum of weights). Hence, the normalized weights construct the probability mass function(*pmf*) at x_i for the discrete random variable X. That is, the *pmf* at x_i represents $Prob(X = x_i)$. Without loss of generalization, we can re-arrange x_1 through x_n based on the ascending order of *pmf*. Then we can obtain the cumulative distribution function(*cdf*) for X. That is, the *cdf* at x_i is the sum of *pmf* at x_1 through x_i. Therefore, the *cdf* at x_i represents $Prob(X \leq x_i)$. Based on these values, we can generate the *pmf* and *cdf* curves by connecting points at each specific value of X.

The motivation of the multi-bit BLASTed linkage is that we select two cutoff values C_1 and C_2 of X so that we can assign 1-bit, 2-bit, and 4-bit coding to the tokens in these three regions respectively (suppose $C_1 < C_2$). Basically we will assign 4-bit coding to those more important tokens, therefore those tokens would fall in the range $[C_2, x_n]$ on the *cdf* curve. There are a variety of potential criteria to select the cutoff values. For instance, we can select C_1 as the 25th percentile or the first quartile, and C_2 as the 75th percentile or the third quartile. This corresponds to the following selection criterion: $Prob(X \leq C_1) = Prob(X \geq C_2) = 0.25$. More generally, given two pre-determined thresholds a_1 and a_2 for the lower and upper boundaries of the cumulative distribution function, we can base on the following criterion to select the cutoff values:

$$Prob(X \leq C_1) = a_1 \text{ and } Prob(X \geq C_2) = a_2$$

The detailed steps for the proposed multi-bit BLASTed linkage is shown in Algorithm 1.

4. EXPERIMENTAL VALIDATION

4.1 Set-up

Platform. All experiments presented here are done on a machine with eight Dual-core 2400MHz AMD Opteron processors and a total of 32G memory. For the evaluation of BLASTed linkage schemes, we used the `blastn` of BLAST v 2.2.13 (linux) from NCBI ftp site[5]. The `blastn` is one of BLAST implementations for searching DNA sequence database. By default, BLAST filters out regions in a query sequence that are frequent in database sequences because those regions are not usually biologically significant. However, since those regions can be important for record

[5]ftp://ftp.ncbi/nih.gov

Name	Data	Error	Domain	# of records	Max # of duplicates	# of queries	# of targets
cora	real	real	citation	1,326	5	98	194
restaurant	real	real	restaurant info.	864	2	111	111
celebrity	real	real	celebrity address	2615	2	276	276
dblp	real	synthetic	citation	5359	5	1,369	3,991
dbgen1	synthetic	synthetic	mailing list	10,159	20	949	9,210
dbgen2	synthetic	synthetic	mailing list	9,947	19	960	8,987
dbgen3	synthetic	synthetic	mailing list	100,327	9	17,842	80,324

Table 3: Summary of data sets.

Algorithm 1: **Multi-bit BLASTed linkage**.

Input : A complete list of tf-idf weights for each token, α_1 and α_2 as two thresholds of *cdf*

Output. tf-idf values C_1 and C_2 to separate 1, 2, and 4-bit coding schemes

```
normalize the weights so that ∑ weight[i] = 1 ;
sort tokens by normalized weights in the ascending order;
/*initialization*/;
n ← number of tokens;
i ← 0, sum ← 0;
/*find C1*/
while i < n and sum < α1 do
    sum ← sum + weight[i], i ← i + 1
endwhile
C1 ← weight[i];
/*find C2*/
j ← n − 1, sum ← 0;
while j ≥ 0 and sum ≤ 1 − α2 do
    sum ← sum + weight[j], j ← j − 1
endwhile
C2 ← weight[j];
return(C1, C2);
```

matching (e.g., year and venue in citation data sets), the [illegible] of BLAST was set to 0.2. Besides, the rest of the parameters were untouched.

`blastp`, BLAST implementation for protein sequences, can use various substitution matrix and each matrix assigns different score for mismatches between different pairs of amino acids. `blastn` uses simpler scoring scheme so that the same score is assigned for mismatches between any two nucleotides. Since it's simple scoring scheme is fit better for the comparison of English strings, we used `blastn` in our experiments.

Even though some parameters to calculate e-value is derived from the observation of biological sequences, e-value statistically indicates how much an alignment is given by of query and database sequences. Thus, e-values of an alignments between a query and each database sequence is considered as a measure of how each database sequence is relatively similar to the query in our context.

Data Sets. Table 3 shows the summary of data sets used in experiments. All three data sets, `cora`, `restaurant`, and `celebrity`, are used as real data sets, containing *real* string data and *real* errors. Both of `cora` and `restaurant` data sets are downloaded from RIDDLE[6] data repository[7]

[6] http://www.cs.utexas.edu/users/ml/riddle/

[7] The celebrity data set was provided to us by Ned Porter at US Census Bureau.

Error Type	Probability
Typo errors exist	0.3
Single typo error	0.6
Multiple typo error	0.4
First name is dropped	0.3
First name is abbreviated	0.7
Middle name is dropped	0.2
Middle name is abbreviated	0.8
Typo errors exist	0.9
Single typo error	0.9
Multiple typo error	0.1
Venue information as a full name	0.4
Pages and month information are dropped	0.5
Numeric information has a typo	0.2

Table 4: Parameter settings used for the `dbgen1` (above) and `dblp` (bottom) data set. Note that probabilities for different error types are independent each other.

First, from the original `cora` data set with 1,296 citation records, we hand-selected 292 records of 98 citations since some of the duplicates are incorrectly labeled and it is unrealistic that duplicates of only a few entities take a large portion of the entire database records like in the original data set. Then, it is extended with 1,034 additional citation records manually collected from the Web.

Second, the `restaurant` data set contains 864 records of restaurant information, such as restaurant name, address, telephone number, and food type that they serve (e.g., Korean, Chinese, Thai), collected from the Fodor's and Zagat's restaurant guides. Each record in the data set has exactly two duplicates.

Third, the original `celebrity` data set has 2764 records of celebrities' addresses with their names. The name of a celebrity is used as a key to distinguish duplicates in the data set. In case of the records with the same address for different celebrity names, only one of them was kept. Among the remaining 2615 records, 276 records are used as query strings and each of them has an exactly one duplicate.

Next, a new citation data set `dblp` was generated using real citation records from similar venues of DBLP. To inject artificial errors, we randomly selected names from similar research domains such as SIGMOD, PODS, and EDBT, and again randomly selected citations published in those venues. Given initial citations, we generated duplicates by introducing typographical errors. For instance, single typo error was generated by inserting a new character, replacing a character by different character, deleting a character, or swapping two characters in a word. For numeric data fields (e.g, journal volumes), we only use insertion and deletion. The detailed parameter settings to generate errors are shown

in Table 4.

Finally, using the data generation tool, DBGen [9], we generated four synthetic data sets, `dbgen1` to `dbgen3`, containing mailing list information with nine attributes such as SSN, first name, middle name, last name, street number, and so on. DBGen permits us to control error rates in records in detail. Both of `dbgen1` and `dbgen2` have around 10,000 records with on average 20 duplicates per record. The third data set, `dbgen3`, have more than 100,000 records for us to study the scalability of the proposed BLASTed linkage methods. For `dbgen1`, the probability to insert an error in the SSN field is set to 0.1 and the probability to change an address completely is 0.7. In `dbgen2`, the probability for a token to have a typo (10% single vs. 90% multiple typos) in person and address fields is subsequently increased to 0.8. The `dbgen3` is similarly made, too.

Evaluation Metrics. To evaluate the efficiency and effectiveness of the proposed BLASTed linkage methods, we mainly use two evaluation metrics – speed and accuracy. In particular, to measure the speed of a method, we use the **running time** excluding any pre-processing steps of sequence translation and database preparation. To measure the accuracy, we use the average precision and recall. Suppose that T denotes a set of true matching records and R denotes a set of records retrieved by an algorithm. Then, we have: precision=$\frac{|R \cap T|}{|R|}$ and recall=$\frac{|R \cap T|}{|T|}$. Since the golden solution set and # of true matching records are available in all of the data sets, in the selection model with top-k answers, we have $R = T = k$, making both precision and recall equivalent. Therefore, hereafter, we simply refer to the accuracy as precision. In addition, we also simulate the threshold model in evaluation by changing the k in top-k. In such a case, we will present the precision-recall graph in the traditional IR sense.

To compare with the performance of BLASTed linkage proposals, we chose three string distance metrics popular in conventional record linkage methods – Jaro, Jaccard, and SoftTFIDF – from the Java-based open-source implementations of the SecondString package [5]. Recent work such as [15] reported good performance of these methods in the context of the citation matching problem. In addition, we used JaroWinkler distance to detect obvious typos. Using symbols like x, y for strings, T_x for all tokens of x, C_x for all characters of x, $CC_{x,y}$ for all characters in x common with y, and $X_{x,y}$ for # of transpositions of character in x relative to y, simple definitions of the four metrics are below: (1) $\mathbf{Jaro}(\mathbf{x},\mathbf{y}) = \frac{1}{3} \times (\frac{|CC_{x,y}|}{|C_x|} + \frac{|CC_{y,x}|}{|C_y|} + \frac{|CC_{x,y}| - X_{CC_{x,y},CC_{y,x}}}{2|CC_{x,y}|})$; (2) $\mathbf{Jaccard}(\mathbf{x},\mathbf{y}) = \frac{|T_x \bigcap T_y|}{|T_x \bigcup T_y|}$; (3) $\mathbf{TFIDF}(\mathbf{x},\mathbf{y}) = \sum_{w \in T_x \cap T_y} V(w,T_x) \times V(w,T_y)$, where $V(w,T_x) = log(TF_{w,T_y}+1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'}(log(TF_{w,T_y}+1) \times log(IDF_w))}}$ (symmetrical for $V(w,T_y)$), where TF_{w,T_x} is the frequency of w in T_x, and IDF_w is the inverse of the fraction of names in a corpus containing w; and (4) $\mathbf{JaroWinkler}(\mathbf{x},\mathbf{y}) = Jaro(x,y) + \frac{max(|L|,4)}{10} \times (1 - Jaro(x,y))$, where L is the longest common prefix of x and y. The SoftTFIDF method is a minor improvement from the TFIDF method to use "soft" token-matching idea [5].

4.2 Comparison with Baseline Approaches

Figure 5(a) shows a per-query execution time computed by dividing the total execution time per data set by the number of queries in the data set. The BLASTed linkage schemes dominated two baseline methods, Jaro and SoftTFIDF, in most of the data sets. Jaccard ran faster than BLASTed linkage methods in `cora`, `restaurant`, and `dblp` data sets.

The execution time of BLASTed linkage schemes is largely affected by the length of sequences to be matched due to the nonlinear nature of its matching algorithm while it is not affected so much by the number of records in the target database as it uses inverted list to prune out unrelated sequences before starting the alignment. Consequently, the difference in the performance two baseline methods we tested are Jaro and SoftTFIDF, and BLASTed linkage methods is much more significant in `dblp` and `dbgen2`, due to the sizes of the data sets, than in `cora` and `restaurant` data sets.

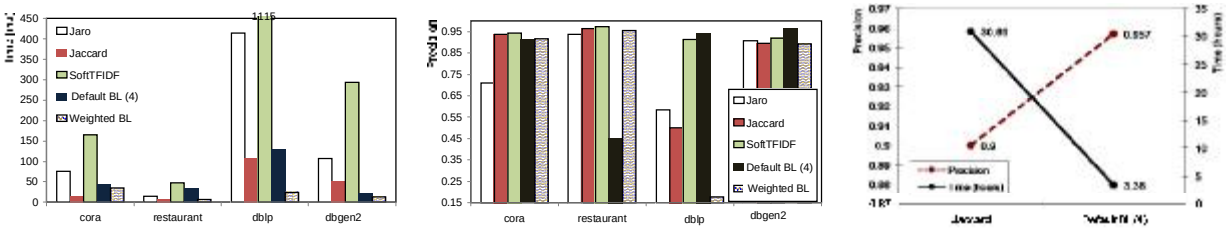
However, the cost of token-based matching methods like Jaccard increases quadratic to the number of records to match. Figure 5(c) highlights the scalability of BLASTed linkage method using the largest `dbgen3` data set, compared to Jaccard. As shown in figure 5(c), in general, Jaccard are fast for small-sized data sets. However, for a substantially large data set, thanks to the fast performance of underlying BLAST algorithm and its optimized implementation, our default BLASTed linkage with 4-bit scheme far outperforms Jaccard by an order of magnitude (3.38 vs. 30.89 hours) while maintaining higher precision (0.957 vs. 0.9).

Record linkage applications deal with large data sets and the scalability of the matching algorithm is one of their primary concerns. We can easily address the problem using BLAST and it is one of the key motivations of this work.

Figure 5(b) shows the precisions of the BLASTed linkage schemes and the baseline approaches measured on the four data sets. Graphs with other data sets show similar patterns and omitted. As shown in the figure, Jaccard and SoftTFIDF showed slightly better precision than the BLASTed linkage schemes in `cora` and `restaurant` data sets. The `cora` data set is particularly challenging because it includes citations of different publications of the same venue information. If the venue of those citations is long, the alignment algorithms can get mislead as the substantial parts of them are matched. Also, there are some numbers of entries that have the same titles and authors but published in different venues. It is virtually impossible to tell the identity of the entries without looking at the actual contents of the papers.

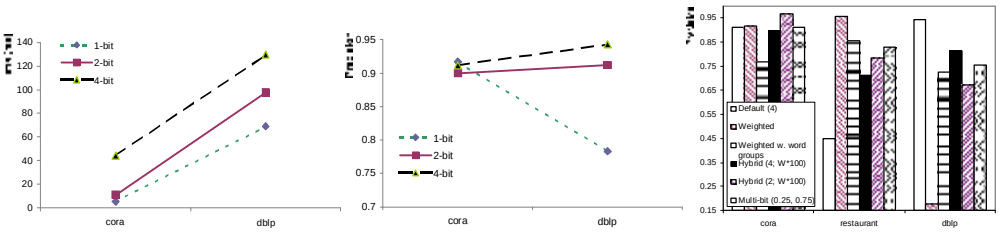
Default BLASTed linkage showed an interesting result on the `restaurant` data set. Compared to its results on the other data set, its precision was very low. This is due to the characteristics of record strings in the `restaurant` data set. The records of `restaurant` data set have restaurant name, address, telephone number, and food type. Among the record fields, the address string takes a large portion of entire record string. The `restaurant` data set include information of restaurants in a few cities in U.S., and thus many records are likely to share a portion of their addresses, such as name of street, city, and area code and so on. If two different restaurant records have large common substrings, default BLASTed linkage scheme easily gets misled to find them as a match.

On the other hand, weighted BLASTed linkage does not suffer from this problem as much as default BLASTed linkage does because it translates words to sequences of "efficient" lengths according to their weights. Common words



(a) (b) (c)

Figure 5: (a)-(b): Comparison of string distance measures vs. the default BLASTed linkage using 4-bit coding and weighted BLASTed linkage. (c)Running time (right Y-axis) and precision (left Y-axis) of the largest dbgen3 data set of 100,327 records.



(a) Running time (ms) (b) Precision (c) Precision

Figure 6: Comparison: (a)-(b): three bit schemes in default BLASTed linkage and (c): six variations of BLASTed linkage methods using three data sets.

such as street names, cities and area codes will get low weights and translated into short sequences, reducing their impacts on alignment. As previously discussed, records of `cora` data set also share common venue names in their record strings. Therefore, precision of weighted BLASTed linkage on the `cora` data set was also better than that of default one. However, in the `cora` data set, the difference between the two was insignificant because the venue information does not take as much portion from the record as the address does in the `restaurant` data set.

Meanwhile, on both of `dblp` and `dbgen2` data sets, default BLASTed linkage showed the best performance. Especially, its performance on `dblp` is worth to note where we introduced a high error rate. The precision of Jaro and Jaccard, which showed good precisions on other data sets, decreased drastically to around 0.5. The precision of weighted BLASTed linkage on `dblp` data set is very low because the errors are introduced to tokens and weighted BLASTed linkage takes the same tokens with typographical errors as different tokens.

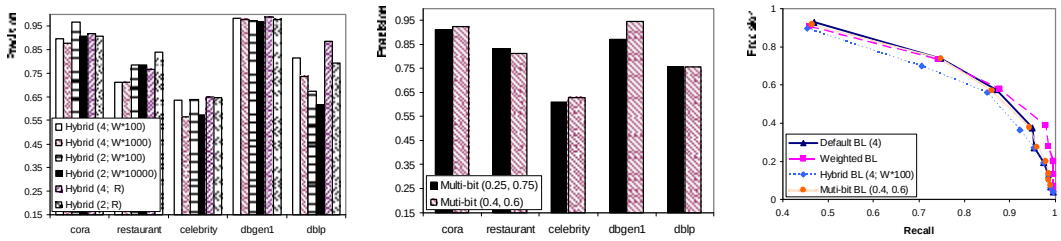
4.3 Comparison of 1/2/4-bit Coding Schemes

Figure 6(a)-(b) presents the execution time and precision of the default BLASTed linkage with three different bit-coding schemes. We compared the default 4-bit coding to two lossy bit-coding schemes of 1-bit and 2-bit codings. Tests performed on `cora` and `dblp` data sets are presented. As can be seen in Figure 6(a), default BLASTed linkage with 4-bit coding is the slowest taking approximately 40 and 130 milliseconds on `cora` and `dblp` data sets, respectively. On the other hand, the 1-bit coding scheme was the fastest taking only less than half of the time that 4-bit coding required. As shown in the figure (b), all three coding schemes showed comparable results on `cora` data set achieving about 0.9 accuracy. On the other hand, on `dblp`, the 4-bit scheme performed slightly better than 2-bit achieving 0.94 as precision while the 1-bit scheme was the worst achieving only 0.78 as precision. The performance degradation of the lossy coding schemas are data set dependent while execution time saving is evident.

4.4 Comparison among BLASTed Linkages

In Figure 6(c), we compared six variations of BLASTed linkage method using three data sets. In the graph, default(4) represents default BLASTed linkage methods with 4-bit coding scheme. And, hybrid (4; W*100) refers hybrid BLASTed linkage method with 4-bit coding scheme and an importance weight, which is the original weight of a token multiplied by 100. Multi-bit (0.25, 0.75) represents multi-bit BLASTed linkage method with a lower cut-off value 0.25 and a higher cut-off value 0.75. The third variant "weighted BL with word group" refers to the application of weighted BLASTed linkage after we coalesce similar words (e.g., typos) into groups using JaroWinkler method.

For `cora` data set, all methods, except the "weighted BL with word group", perform well, above the precision of 0.9. The hybrid BLASTed linkage methods with 2-bit coding scheme and an importance weight of the original weight multiplied by 100 outperforms other methods. The real data/error of `restaurant` data set, weighted BLASTed linkage shows the best precision. Because the data set has high chance that a large portion of the tokens in two records are common, weighted BLASTed linkage methods improves the precision of default BLASTed linkage method as considering word weight. Finally, for the synthetic data set `dblp`, default BLASTed linkage methods outperform the rest. It is



(a) (b) (c)

Figure 7: (a) Comparison among four variations of hybrid BLASTed linkage methods using five data sets. (b) Comparison between multi-bit BLASTed linkage methods with two cutoff parameters using five data sets. (c) Precision-recall graphs of three variations of BLASTed linkage methods using `cora` data set.

interesting that the precision of weighted BLASTed linkage method, the best method for `restaurant` data set, is very low due to typographical errors introduced in the data set.

Overall, there is no clear winner across three data sets as inferring that each BLASTed linkage method performs differently depending on the characteristics of each data set, such as the number of tokens shared in different records and types/degrees of errors in the data set. Multi-bit BLASTed linkage method does not outperform other BLASTed linkage methods in all data sets, however it seems to provide robust performance in overall. The performance of hybrid and weighted BLASTed linkage with word group is relatively consistent. Also, it seems that Default and weighted BLASTed linkage methods are affected by the characteristics of a data set the most.

4.5 Hybrid vs. Multi-bit BLASTed linkage

As shown in Figure 7, the performance of hybrid BLASTed linkage method is tested on five data sets as using six combinations of different bit coding schemes and types of importance weight. Either of a token weight or a token rank is used as importance weight for each token. The original token weight is multiplied by either of 100 or 10,000 before the weight is translated to DNA sequence. As multiplying larger value, we can increase the proportion of weight information in DNA sequence of a records. Also, as using larger number-bit scoring scheme, the proportion of lexical meaning information of each token is increased.

With the same importance weight, hybrid BLASTed linkage methods with 4-bit coding scheme outperforms the methods with 2-bit coding schemes in `dbgen1` and `dblp`. However, the result is the opposite in `cora` and `restaurant` data sets. In `celebrity` data set, different coding schemes do not affect the performance of hybrid BLASTed linkage. Next, given the same bit coding scheme, hybrid BLASTed linkage method with smaller importance weight shows better precision than that with larger importance weight in all data sets except for `restaurant` data set. Different importance weight does not change the performance of hybrid BLASTed linkage methods in `restaurant` data set.

Overall, hybrid BLASTed linkage method with larger number-bit coding scheme and smaller importance weight works better than other methods in both of `dbgen1` and `dblp`. For `celebrity` data set, hybrid BLASTed linkage with largest importance weight show better precision. Contrast to `celebrity` data set, which bit coding scheme is used is more important in `restaurant` data set and hybrid BLASTed linkage smaller value bit coding scheme outperforms that with larger value bit coding scheme. For `cora` data set, it is better to use hybrid BLASTed linkage with smaller value bit coding scheme and smaller importance weight.

To see the effect of different cutoff values for multi-bit BLASTed linkage method, two different cutoff values such as (0.25, 0.75) and (0.4, 0.6), are used. Figure 7 shows the result of the two variations using five data sets. We cannot conclude that which variation clearly outperform the other. It is possible to find optimal cutoff values for selection criterion of bit coding schemes in clusters with different weights may depend on data sets.

4.6 Discussion

In this section, we will discuss about variations of BLASTed linkage method to answer the following questions: (1) *Which variation of BLASTed linkage method would work optimal on the data set with particular characteristics?* (2) *Which variation of BLASTed linkage method performs consistently in overall?*

Default BLASTed linkage method performs consistently well on the data sets with both of low and high error rate, except for the data sets with many records sharing common tokens as much as a significant portion of the entire records. On the other hands, weighted BLASTed linkage method performs well with such data sets, as well as those with low error rate. For the data sets with relatively high error and the records sharing many common tokens, either of hybrid and multi-bit BLASTed linkage methods would work well. For the data set with high error rate, hybrid BLASTed linkage method with 4-bit coding scheme and small importance weight performs better than hybrid method with 2-bit coding scheme and large importance weight. However, for the data set with low error rate, hybrid method with 2-bit coding scheme and large importance weight outperforms. Multi-bit BLASTed linkage method seems perform consistently well for all kinds of data sets. It is true that we still need to decide the proper cutoff values for selection criterion of bit coding schemes, but the performance of multi-bit BLASTed linkage method does not change much with reasonable cutoff values as shown in figure 7(b).

The precision-recall graph in figure 7(c) is generated with precision and recall of three variations of BLASTed linkage methods as increasing k, which is the number of answers returned (e.g. top-k answers). With respect to recall, precision of default and hybrid BLASTed linkage methods is changed as following very similar curve. As recall is in-

creased, precision of multi-bit BLASTed linkage method is dropped faster than default and hybrid BLASTed linkage methods. However, the difference is not significant. Therefully selected BLASTed linkage method which is suitable for a certain data set can guarantee better performance on the data set. Either of hybrid and multi-bit BLASTed linkage methods perform robust for any kind of data sets despite.

5. RELATED WORK

The general linkage problem has been known as various names – record linkage (e.g., [7, 3]), identity uncertainty (e.g., [16]), merge-purge (e.g., [9]), citation matching (e.g., [15]), object matching (e.g., [4]), entity resolution (e.g., [18]), authority control (e.g., [21]), and approximate string join (e.g., [8, 4]) etc. Readers are referred to excellent survey papers (e.g., [22, 6]) for the latest development of the linkage problem. To our best knowledge, none of these existing works attempted to solve the linkage problem using bioinformatics framework as we did it using BLAST. In terms of distance measures to use in comparing records, Cohen et al. [5] have compared the efficacy of various string distance metrics including Jaro and Jaro-Winkler. [15] conducted an in-depth study on the blocking issue of the linkage problem in digital library context. On the other hand, in terms of scalability issue, people have proposed the blocking technique [11], computationally efficient distance function [14], or parallel linkage [12].

To our best knowledge, there have been very few attempts to use BLAST for applications outside of gene sequence alignment. First, [2] borrows the idea of multiple sequence alignment from BLAST in building a mapping dictionary, that is a lexicon of elementary semantic expressions and corresponding natural language realizations. Second, [13][8] uses BLAST to detect gene and protein names from journal articles by viewing an entire article as a query sequence and a set of known gene and protein terms as a database of sequences to match. Finally, [10] employs BLAST in the citation field segmentation problem – i.e., split a unsegmented citation string into known set of fields such as author, title, venue, and year. In the same spirit of these works but for the first time, we propose to use BLAST to solve the *linkage* problem. Moreover, we devise/evaluate four effective variations to transform regular string data to DNA sequences.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented the novel idea of applying BLAST, one of the most popular sequence alignment algorithms in Bioinformatics, to the (record) linkage problem. To handle various cases of the linkage problem, we proposed four variations of the BLAST based linkage schemes, and show their efficacy using both real and synthetic data sets. Overall, our proposed BLAST based schemes show promising results with good combination of accuracy and speed, compared to conventional string matching methods such as Jaro, Jaccard, and SoftTFIDF. Since BLAST has a wealth of supporting algorithms, implementations, and tools in its user community, our proposal enables users of the linkage problem to have an immediate access to rich resources.

Many directions are ahead. First, in this paper, we only focused on exploiting the basic version of BLAST for the linkage problem. Applying the very same idea using more recent and advanced BLAST versions (e.g., PSI-BLAST, MegaBLAST) is an interesting extension to try. Second, many domain specific linkage schemes have been proposed. Comparing the performance of our BLASTed linkage schemes against other state-of-the-art (but less generic) solutions would be able to help improve the weakness of our ideas further.

[8] To our best knowledge, [13] is the first article mentioning the idea of transforming an English alphabet to DNA sequences of `A`, `C`, `G`, and `T`.

7. REFERENCES

[1] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. "Basic Local Alignment Search Tool". *J. Mol. Bio.*, 1990.

[2] R. Barzilay and L. Lee. "Bootstrapping Lexical Choice via Multiple-Sequence Alignment". In *Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[3] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. "Adaptive Name-Matching in Information Integration". *IEEE Intelligent System*, 18(5):16–23, 2003.

[4] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. "Robust and Efficient Fuzzy Match for Online Data Cleaning". In *ACM SIGMOD*, 2003.

[5] W. Cohen, P. Ravikumar, and S. Fienberg. "A Comparison of String Distance Metrics for Name-matching tasks". In *IIWeb Workshop held in conjunction with IJCAI*, 2003.

[6] A. Elmagarmid, P. Ipeirotis, and V. Verykios. "Duplicate Record Detection: A Survey". *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 19(1):1–16, 2007.

[7] I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 1969.

[8] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins in an RDBMS for Web Data Integration". In *Int'l World Wide Web Conf. (WWW)*, 2003.

[9] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.

[10] I.-A. Huang, J.-M. Ho, H.-Y. Kao, and S.-H. Lin. "Extracting Citation Metadata from Online Publication Lists Using BLAST". In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, 2004.

[11] R. P. Kelley. "Blocking Considerations for Record Linkage Under Conditions of Uncertainty". In *Proc. of Social Statistics Section*, pages 602–605, 1984.

[12] H. Kim and D. Lee. "Parallel Linkage". In *ACM CIKM*, Lisbon, Portugal, Nov. 2007.

[13] M. Krauthammer, A. Rzhetsky, P. Morozov, and C. Friedman. "Using BLAST for Identifying Gene and Protein names in Journal Articles". *Gene*, 259, 2000.

[14] B.-W. On, N. Koudas, D. Lee, and D. Srivastava. "Group Linkage". In *IEEE ICDE*, Istanbul, Turkey, Apr. 2007.

[15] B.-W. On, D. Lee, J. Kang, and P. Mitra. "Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2005.

[16] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. "Identity Uncertainty and Citation Matching". In *Advances in Neural Information Processing Systems*. 2003.

[17] H. Rangwala, E. Lantz, R. Musselman, K. Pinnow, B. Smith, and B. Wallenfelt. "Massively Parallel BLAST for the Blue Gene/L". In *High Availability and Performance Computing Workshop (HAPCW)*, Santa Fe, NM, 2005.

[18] S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM KDD*, 2002.

[19] W. Shen, X. Li, and A. Doan. "Constraint-Based Entity Matching". In *AAAI*, 2005.

[20] T. Smith and M. Waterman. "Identification of Common Molecular Subsequences". *J. Mol. Biology*, 147, 1981.

[21] J. W. Warnner and E. W. Brown. "Automated Name Authority Control". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, 2001.

[22] W. E. Winkler. "The State of Record Linkage and Current Research Problems". Technical report, US Bureau of the Census, Apr. 1999.

EARL: an Evolutionary Algorithm for Record Linkage

Francesco Cusmai
SAPIENZA – Univ. Roma, ITALY
cusmai.francesco@gmail.com

Carola Aiello
SAPIENZA – Univ. Roma, ITALY
carola.aiello@dis.uniroma1.it

Monica Scannapieco
ISTAT and SAPIENZA – Univ. Roma, ITALY
scannapi@istat.it

Tiziana Catarci
SAPIENZA – Univ. Roma, ITALY
catarci@dis.uniroma1.it

ABSTRACT

Record linkage is an important problem in all data integration contexts that assume to be valid the hypothesis of data sources affected by errors.

In this paper we formulate the record linkage problem as a multiobjective optimization problem and we propose an algorithm for solving it based on evolutionary computation. This computational paradigm has a stochastic, adaptive and intrinsically parallel nature that makes it particularly suitable for on-the-fly data integration contexts.

Experiments on real datasets proved the effectiveness and efficiency of this novel approach.

1. INTRODUCTION

The Record Linkage (RL) problem arises in all application scenarios in which a data integration activity must be performed. Given two (or more records) RL has the purpose of comparing them on some common attributes and deciding if they are a match or not, on the basis of the result of such a comparison. The complexity of the problem resides in the lack of identifiers for the sources to be integrated and in the presence of errors in the records to be matched.

The record linkage problem has been studied for more than five decades. More specifically, record linkage techniques can be classified into: probabilistic, knowledge-based and empirical [2]. The main reference for the probabilistic RL literature is the Fellegi and Sunter model [8]; many different methods have been proposed to estimate the parameters of such a model, most of them based on the Expectation-Maximization method [12, 22]. One problem of the probabilistic approaches is related to the estimation of the parameters of the Fellegi and Sunter model with large data sets [23]. In knowledge-based techniques (e.g. [17]) the RL process is based on the use of rules. These rules represent the domain knowledge that is extracted from the data sources to be matched, and reasoning strategies are applied to make the process more effective. These techniques are typically based on the usage of training sets to learn rules, so they are appropriate for contexts where training sets are available and can be easily used. Empirical techniques (see [7] for a survey) focus mainly on the search space reduction problem and are typically used in conjunction with techniques of the cited paradigms. Though there is a rich scientific production for the RL problem, the provided solutions are not immediately applicable to a new data integration paradigm that is currently emerging.

The current scenario for accessing separate data sources in a unified way presents two extreme situations. On the one hand, there are traditional data integration systems successfully working, such as Expedia. On the other hand, there are search engines, like Google, that access extensive parts of the Web, but have an inherent page-based view of data. It is becoming more and more important for several applications to have systems that: (i) do not have the complexity and the limitations in terms of scale and rigidity of traditional data integration systems; (ii) can do more than the page-based access style of search engines.

In order to face the challenges mentioned above, there is the urgent need of a new generation of systems that must be able to do on-the-fly data integration, that is to perform integration with specific and well-defined features: quick set-up and access time, large scale and dynamicity.

Given such strict requirements, new RL strategies should be worked upon. For instance, the usage of probabilistic techniques for RL has the problem of dealing with large datasets. Knowledge-based techniques heavily rely on training set data and on stable configuration of the sources to be integrated.

To overcome these limitations, we propose to solve record linkage in on-the-fly integration scenarios by using a completely novel approach, based on evolutionary computation. Evolutionary algorithms have all the features desired for this specific setting: they are stochastic, adaptive and intrinsically parallel search algorithms. Indeed, the stochastic feature and the intrinsic parallelism can help to face problems deriving from large scale; adaptiveness is very important for the dynamicity and openness features.

More specifically, in this paper, we first formulate the record linkage problem as a multiobjective optimization problem. Then, we extend an evolutionary algorithm named NSGA-II in order to solve our problem. We experimentally show that our algorithm has good performance, both in terms of effectiveness and efficiency.

The rest of the paper is organized as follows. After some background definitions, provided in Section 2, Section 3 describes two formulations of the record linkage problem and

VLDB '08, August 24-30, 2008, Auckland, New Zealand

Section 4 describes the details of the proposed algorithm. Finally, we present our experimental results in Section 5 and we draw some conclusions in Section 7. The related work is presented in Section 6.

2. DEFINITIONS

Before presenting the details of our algorithm we provide some background definitions. In general, multiobjective optimization problems can be defined in the following way [4, 3]:

$$Min/Max \quad f_m(x) \quad m = 1, 2, ..., M \tag{1}$$

$$g_j \geq 0 \quad j = 0, ..., J \tag{2}$$

$$h_k = 0 \quad k = 1, ..., K \tag{3}$$

$$x_i^L \leq x_i \leq x_i^U \quad i = 1, ..., n. \tag{4}$$

Where f is the function to optimize, *(2)* and *(3)* represent constraints to be satisfied and *(4)* is the interval of values that x_i can assume.

The differences between multiobjective optimization problems and the single-objective ones are:

- M objectives instead of a single one;
- beside the n-dimensional space of the admissible solutions (which is also present in single-objective optimization problems), there is a further M-dimensional space whose coordinates correspond to the M objectives of the problem;
- generation of a set of solutions.

In a multiobjective optimization scenario a different definition of optimum is needed. In the following we introduce the notion of Pareto optimum that is the one mainly adopted in these cases.

Definition - Pareto Optimality *A solution $x \in \Omega$ is Pareto optimal in Ω if and only if there is no $y \in \Omega$: $v = F(y) = f_1(y), f_2(y), ..., f_k(y)$ dominates $u = F(x) = f_1(x), f_2(x), ..., f_k(x)$.*

To state if a solution x is better than a solution y we refer to the following definition:

Definition - Pareto Dominance *A vector $u = (u_1, ..., u_k)$ dominates a vector $v = (v_1, ..., v_k)$ ($u \preceq v$) if and only if $\forall i \in \{1, .., k\} \quad u_i \leq v_i$ and $\exists i \in \{1, .., k\}$ $u_i < v_i$.*

With this ordering relation between the solutions an optimal set will be defined in the following way:

Definition - Pareto Optimal Set *Let us consider an optimization problem where $F(x)$ is the function to be optimized, the Pareto Optimal Set $\mathcal{P}$ is: $\mathcal{P} = \{x \in \Omega \mid \nexists y \in \Omega \quad F(y) \preceq F(x)\}$.*

For each decision variable belonging to the Pareto Optimal Set there is a correspondent vector in the space of the objective function as stated by the following definition:

Definition - Pareto Front *Let us consider an optimization problem with an objective function $F(x)$ and a Pareto Optimal Set $\mathcal{P}$, the Pareto Front $\mathcal{F}$ is defined as:*
$\mathcal{F} = \{u = F(x) \mid x \in \mathcal{P}\}$.

Multiobjective optimization problems can be solved with evolutionary algorithms [1]. Evolutionary computation is a paradigm inspired by the biological evolution process that is ruled by natural selection [13]; it includes techniques of genetic algorithms, evolution strategies and evolutionary programming [10]. The use of evolutionary algorithm to solve multioptimization problems is mainly due to the population-based nature of these algorithms which allows the generation of several elements of the Pareto Optimal Set in a single run. A formal definition for an evolutionary algorithm is provided in the following.

Definition - Evolutionary Algorithm *An evolutionary algorithm is defined as an 8-tuple*
$EA = (I, \Phi, \Omega, \mu, \lambda, \iota, \Psi, \Upsilon)$
where:

- $I = A_x \times A_s$ is the space of the individuals with A_x and A_s arbitrary sets.
- $\Phi : I \longmapsto \mathbb{R}$ denotes the fitness function which assigns real values to the individuals.
- $\Omega = \{\omega_{\theta_1}, ..., \omega_{\theta_z} \mid \omega_{\theta_i} : I^{\lambda} \rightarrow I^{\lambda}\} \cup \{\omega_{\theta_{z+1}} : I^{\mu} \rightarrow I^{\lambda}\}$ is a set of probabilistic genetic operators ω_{θ_i}, each one controlled by specific parameters contained in $\theta_i \in \mathbb{R}$. λ is the number of *son-individuals* and μ is the number of *father-individuals*.
- $s_{\theta_i} : (I^1, .., I^{\mu+\lambda}) \rightarrow I^{\mu}$ denotes the selection operator that can change the number of individuals from λ to $\mu + \lambda$ where $\lambda = \mu$ is allowed. A further set of parameters Θ_s can be used by the selection operator.
- Finally $\iota : I^{\mu} \mapsto \{true, false\}$ is an ending condition for the evolutionary algorithm and the transition function $\Psi : I^{\mu} \mapsto I^{\mu}$ describes the transformation process of a population P into the next one by applying the genetic operators and the selection:
  - $\Psi = s :: \omega_1 :: ... :: \omega_j :: \omega_n$
  - $\Psi(P) = s_{\theta_i}(Q \bigcup \omega_{i_1}(...(\omega_{i_j}(\omega_n(P)))...))$
  - $\{i_1, ..., i_j\} \subseteq \{1, ...z\} \quad Q \in \{\emptyset, P\}$

3. PROBLEM FORMULATION

Let us consider two files, say A and B, of dimension n and m respectively. Without loss of generality we assume that $n \leq m$.

In our work we formulate the RL problem as a multiobjective optimization problem, according to two distinct formulations, detailed in the following.

FORMULATION I

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} F(y_i^A, y_j^B) \cdot x_{ij} \tag{5}$$

$$\sum_{j=1}^{m} x_{ij} \leq 1 \tag{6}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \tag{7}$$

$$F(y_i^A, y_j^B) = \begin{bmatrix} f_1(y_{i1}^A, y_{j1}^B) \\ f_2(y_{i2}^A, y_{j2}^B) \\ \cdot \\ \cdot \\ \cdot \\ f_k(y_{ik}^A, y_{jk}^B) \end{bmatrix} \quad (8)$$

$$x_{ij} \in \{0, 1\}$$

Variables x_{ij} are equal to 1 when record j of file B has been assigned/matched to record i of file A and 0 otherwise.

The function $F(y_i^A, y_j^B)$ calculates the distances between fields of record i of file A and fields of record j of file B and it is composed by the functions $f_1(y_{i1}^A, y_{j1}^B), ..., f_n(y_{in}^A, y_{jn}^B)$. These functions can be chosen among a set of comparison functions between strings such as Edit Distance, Jaro and others [6]. Note that a different function can be used for comparing different fields.

Constraints (6) and (7) assure that the match is one to one. Specifically, a record of the file A can be matched with at most one record of the file B and viceversa.

The goal is to maximize the total values for comparison functions.

Example.

Let us suppose to run a RL procedure between table 1 and table 2, shown in the following. As a first step we focus on the common attributes of file A and file B thus obtaining files A' and B' shown in tables 3 and 4.

Table 1: File A

N	Surname	Address	Income
1	Smith	Wasington str.	20000
2	Smith	Lincoln str. 16	10000
3	White	Lincoln str. 90	50000

Table 2: File B

N	Address	Name	Sex	Surname
1	Washington str. 16	Antony	M	Smith
2	Lincoln str.	Lucy	F	Smit
3	Lincoln str. 90	Mary	F	White
4	Kennedy str. 90	John	M	Whit

Table 3: File A'

N	Surname	Address
1	Smith	Wasington str.
2	Smith	Lincoln str. 16
3	White	Lincoln str. 90

Table 4: File B'

N	Surname	Address
1	Smith	Washington str. 16
2	Smit	Lincoln str.
3	Whit	Lincoln str. 90
4	White	Kennedy str. 90

The problem formulation for this input is:

$$\max \sum_{i=1}^{3} \sum_{j=1}^{4} F(y_i^{A'}, y_j^{B'}) \cdot x_{ij}$$

$$\sum_{j=1}^{4} x_{ij} \leq 1$$

$$\sum_{i=1}^{3} x_{ij} \leq 1$$

$$F(y_i^{A'}, y_j^{B'}) = \begin{bmatrix} EditDistance_1(y_{i1}^{A'}, y_{j1}^{B'}) \\ EditDistance_2(y_{i2}^{A'}, y_{j2}^{B'})) \end{bmatrix}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, .., 3 \quad j = 1, .., 4$$

Notice that for both *Surname* and *Address* we have chosen the edit distance as comparison function.

An optimal solution for this case is obtained with the following assignment of the variables:

$$x_{11} = 1 \quad x_{22} = 1 \quad x_{33} = 1$$
$$\forall ij \quad (1 \leq i \leq 3 \wedge 1 \leq j \leq 4 \wedge i \neq j) \rightarrow x_{ij} = 0.$$

Formulation I does not take into account any threshold value when comparing different record fields on the basis of similarity functions. However, this is a common input to several record linkage procedures; therefore, we include these thresholds in a second problem formulation, detailed in the following.

FORMULATION II

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} (F(y_i^A, y_j^B) \cdot x_{ij}) \cdot t_{ij} \quad (9)$$

$$\sum_{j=1}^{m} x_{ij} \leq 1 \quad (10)$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \quad (11)$$

$$F(y_i^A, y_j^B) = \begin{bmatrix} f_1(y_{i1}^A, y_{j1}^B) \\ f_2(y_{i2}^A, y_{j2}^B) \\ \cdot \\ \cdot \\ \cdot \\ f_k(y_{ik}^A, y_{jk}^B) \end{bmatrix} \quad (12)$$

$$\mathcal{V} = \begin{bmatrix} \nu_1 \\ \cdot \\ \cdot \\ \cdot \\ \nu_k \end{bmatrix} \quad (13)$$

$$t_{ij} = 1 \quad iff \quad F(y_i^A, y_j^B) \geq \mathcal{V} \text{ and } x_{ij} = 1 \quad (14)$$

$$\nu_i \in [0, 1]$$

$$t_{ij} \in \{0, 1\}$$

$$x_{ij} \in \{0, 1\}$$

The objective of this problem formulation is the maximization of the global similarity values of those match that satisfy the condition $F(y_i^A, y_j^B) \geq \mathcal{V}$ and $x_{ij} = 1$ where $\mathcal{V}$ is a vector of similarity thresholds that a match must exceed or at least be equal in order to be considered a true match. For this second formulation of the problem we developed an algorithm that uses a specific mutation genetic operator in order to focus the search. The genes of the individual of the population that is submitted to mutation are changed in a selective way. The mutated genes are those that correspond to matches that do not satisfy the condition $F(y_i^A, y_j^B) \geq \mathcal{V}$ and $x_{ij} = 1$.

4. EARL: AN EVOLUTIONARY ALGORITHM FOR RECORD LINKAGE

In this section we describe the algorithm we propose for record linkage, called EARL (Evolutionary Algorithm for Record Linkage). The aim of the algorithm is to generate a matching of records where the total distance between each attribute selected for the match is maximized, according to the given problem formulations.
The algorithm is composed by the phases shown in Figure 1, which are described in the following.

4.1 Phase I: Initialization and Creation of the First Population

The parameters that are input to EARL are:

- The dimension of the initial population, the number of generations and a parameter, named x, which is a negative integer that supports the creation of individuals (as better detailed in the following). All these parameters are set on the basis of the results we have obtained from tests on real datasets, detailed in Section 5.

- Matching attributes, that is those attributes of the input files that will be actually compared for taking the matching decision. These attributes are chosen by a domain expert.

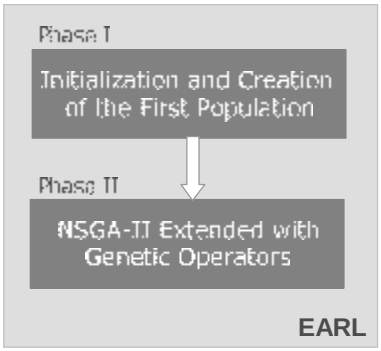


Figure 1: The phases of EARL

In the following, we will indicate as *donor file*, the file which has the highest number of records; we will indicate as *receiving file* the other one. The donor and receiving files are read in two multidimensional arrays, with a number of rows equal to the records of each file and a number of columns equal to the number of attributes selected for the match. The first population is generated in a random way starting from the files which are input to the record linkage procedure. Let m be the dimension of the donor file and n be the dimension of the receiving file. Each individual of the initial population, say v, is represented by an array of objects with dimension n. Each object is a vector of k values corresponding to values of the comparison function function for each attribute in case of a match, it is vector of zero otherwise. The following pseudocode shows how each individual is created.

for i=1 to n
    z=`randomGen(x,m)`

    if z<0
        $[v_{i1} \dots v_{ik}] = [0, \dots, 0]$

    else if z=q>0

        $[v_{i1} \dots v_{ik}] = [f_1(y_{i1}^{don}, y_{q1}^{rec}), \dots, f_k(y_{ik}^{don}, y_{qk}^{rec})]$

The function `randomGen` generates a random value between `x` and `m`, where `x` is a parameter set in dependence of the value of `m` (see Section 5); the idea is to select randomly the index of the record of the donor file to match (or not) with the current index of the record of the receiving file. The variable z has a negative value in the case of a decision of not assigning a match for the corresponding records which have been compared; it has a positive value otherwise.

In order to satisfy the RL one-to-one constraint, when creating the new individual a preventive check is performed to avoid the creation of inadmissible solutions.

4.2 Phase II: NSGA-II extended with Genetic Operators

4.2.1 NSGA-II

NSGA-II is a multi-objective evolutionary algorithm that sorts the solutions according to the Pareto non-dominance principle. The steps of NSGA-II are the followings [4, 3, 5]:

$NSGA - II(N, g, f_k(x_k))$
N number of individuals, g number of generations, $f_k(x_k)$ function to be optimized

1. Initialization of a population P_1 of size N
2. Evaluation of the individuals for each objective
3. Assignment of a rank to each individual in function of a ranking based on the Pareto dominance principle
4. Generation of individuals from P_1 in order to obtain a population P_1' of size $2N$
   (a) Extraction of individuals from P_1 with a selection
   (b) Crossover and mutation of the extracted individuals
5. *for* $i = 1$ *to* g

   For each individual of population P_i' with $i > 1$
   (a) According to the Pareto dominance principle, assignment of a rank to each individual and conforming partition of the population into non-dominated fronts F_k
   (b) Creation of a new population P_{i+1} of size N from P_i'
   for $j = 0$ *to* $j < k$ $F_j.size < N - P_{i+1}.size$
   sorting the solutions of F_j by the so-called crowding distance [1]
   $P_{i+1} = P_{i+1} \cdot . F_j$
   (c) If $N - P_{i+1}.size > 0$
   sorting of the solutions of F_j according to the crowding distance
   adding of the first $N - P_{i+1}.size$ of F_J to P_{i+1}
   (d) Creation of P_{i+1}'' through selection, crossover and mutation on P_{i+1}

The algorithm creates a population of individuals, it sorts each individual according to its level of non-domination then it applies the evolutionary operators to the population. The favorite individuals for the new population are those with the best rank and with the best crowding distance (elitism).

4.2.2 Input Parameters to NSGA-II

The algorithm NSGA-II is initialized in the following way:

- N is the initial population where each individual represents a linkage between the receiving file and the donor file;
- g is the number of generations;
- the function to be optimized is:

$$F(v) = \sum_{i=1}^{n} [v_{i1}, \ldots, v_{ik}] = [v_1, \ldots, v_k] \qquad (15)$$

  where v_1 denotes (a value for) the global similarity between the strings of the field that corresponds to the first matching attribute of the record of the receiving file and of the donor file of the individual v.
- the genetic operators defined in the following section.

4.2.3 Definition of the Genetic Operators

In this section we describe the genetic operators used in EARL. The selection operator is binary tournament, as typically used in NSGA-II.

The binary tournament genetic operator selects the individuals of the population in the following way:

1. a sorted array whose length is equal to the dimension of the population is created, each field of the array contains an integer that represents an individual of the population, the array can contain the same elements;
2. it takes the individual i and the individual $i + 1$, selected between the following:
   - the individual with the best rank;
   - the individual with the best crowding distance, when having the same rank;
   - the individual is randomly selected, when having the same rank and the same crowding distance.

The crossover and mutation operators have been newly defined in order to fit our specific problem.
Our crossover operator takes as input two individuals and returns two individuals built with the best genes of the starting individuals. To perform this kind of crossover the genes of the starting individuals are compared one by one, if there exists a gene that dominates (according to the Pareto Dominance principle) the other, this gene is selected to create the new individual, otherwise, if none of the individuals dominate another one, the choice will be random.
Our mutation operator changes those genes that correspond to values of the similarity function that will probably lead to false-matching. Specifically the similarity functions assume values between 0 and 1 and if the value computed between any two attributes of two records is 0 then with a high probability the two records don't represent the same entity.

The genes of the individual submitted to mutation are changed in part according to the described mutation operator and in part randomly. The described mutation is mainly performed in the first generations and becomes less frequent in the next generations when the ratio between non-good genes and the total genes is less significant. On the other hand the random mutation, that is always applied to the same part of the population during all the generations, avoids the risk of a too premature convergence and the risk of falling into local maxima of the function.
In order to satisfy the RL one-to-one constraint, when creating the new individual a repairing action is performed.

[1] The crowding distance is computed for each objective i in the following way:
- sorting of the solutions according to the objective i;
- computation of the distance of the individual s as the difference between the values of f_i assumed by the previous and by the next individual of s, this value is normalized with the difference that the first and the last individuals assume for objective i;
- the crowding distance of individual s will be the sum of these distances obtained for each objective i.

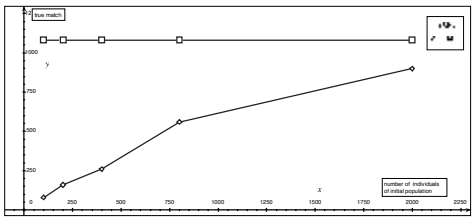


Figure 2: Number of true matches vs. dimension of initial population

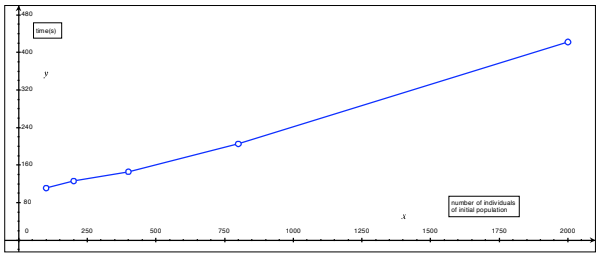


Figure 3: Execution time vs. dimension of initial population

5. EXPERIMENTS

This section describes a set of experiments performed with the objective of testing the effectiveness and efficiency of EARL.

We have used real data sets about personal data related to Rome citizens, owned by an Italian public administration. Specifically, we have run experiments on two datasets that were populated by different processes, each of size of about 8000 records. On these datasets, in the following called dataset A and dataset B, a record linkage activity had already been performed in the context of a project lasted several months and involving both deterministic and probabilistic procedures. A significant human intervention was also necessary in order to decide the status of matching of records. We used these already available results as a benchmark for evaluating the effectiveness of EARL.

As far as the technological platform, we ran experiments on a PC with CPU of 2 Ghz and RAM of 1 GB, with OSX operating system. EARL was developed in Java. A tool called jmetal [14] supported the development of the evolutionary algorithm, and an open source java library was used for string comparison functions [19].

5.1 Effectiveness and Efficiency Experiments

In order to better study the behavior of EARL, we extracted 3 subsets of data from datasets A and B, respectively of sizes 300, 600, and 1000.

First of all we studied how to set up the dimension of the initial population.

Figure 2 shows how the number of true matches increases with the dimension of the initial population for the two problem formulations we have provided. Specifically, algorithm EARL 1 corresponds to the first problem formulation: the curve consistently grows up with number of individuals. Algorithm EARL 2 corresponds to the second problem formulation, in which thresholds on the matching attributes have all been fixed to 0.8. As it is shown, the EARL 2 curve is constant when varying the number of individuals of the initial population, which is a quite interesting finding. In order to understand the effect on the time behavior of this result, we have studied the execution times, when varying the number of individuals of the population. In Figure 3, we report the execution time for the dataset of 1000 records: obviously the time increases with the dimension of the initial population, though quite slowly, confirming the substantial independence of the EARL 2 algorithm from the dimension of the initial population.

A second set of experiments was made in order to consider how to fix the number of generations. Such experiments allowed to study the number of *evaluations*, defined as the product between the number of individuals and the number of generations. The experiments showed:

- A few number of generations was necessary for the convergence of EARL 1. [2]
- how to fix the number of generations of EARL 2, in dependence of the behavior of the number of evaluations when varying the dimensions of the input datasets.

A third set of experiments allowed us to choose a value for the `x` parameter (see Section 4). We have observed that the parameter has an impact on how fast the algorithm converges, and hence on effectiveness and efficiency. More specifically, high values of `x` leads to a low effectiveness, remaining constant the number of generations and the dimension of the population. Hence, a rule of thumb for the choice of `x` is to consider quite low values, between 2% and 5% of the dataset with the highest dimension.

Finally, in Figure 4, we report the effectiveness performance of EARL 1 and EARL 2 that must be compared to the real performance figures we had for our benchmark data. The number of true matches is reported for EARL 1, EARL 2 and the real data. We can notice that:

- the performance of EARL 1 are quite good for datasets of 300, 600 and 1000 records, while become dramatically low for the 8000 record data set. This is mainly due to the fact that for such data set it was not possible, due to memory limitations of the experimental environment, to increase the number of individuals of the initial population in order to have better performance.
- the performance of EARL 2 (which are independent on the number of individuals of the initial population) are instead very good, even when compared with exact methods for record linkage, being the percentage of the true matches always higher than 80% of the real true matches with peaks of more than the 90%.

In synthesis, algorithm EARL 1 is really independent on the data, and does not need to fix parameters like threshold values that require an accurate analysis of the data at hand. On the other hand it can have performance that are not

[2]Actually we verified that the behavior of EARL 1 can lead to a premature convergence to local maxima independent on the number of generations.

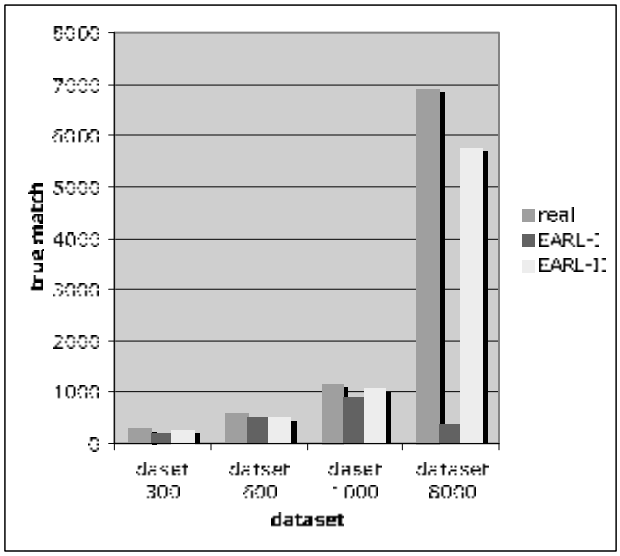


Figure 4: True matches for EARL 1, EARL 2 compared the real true matches

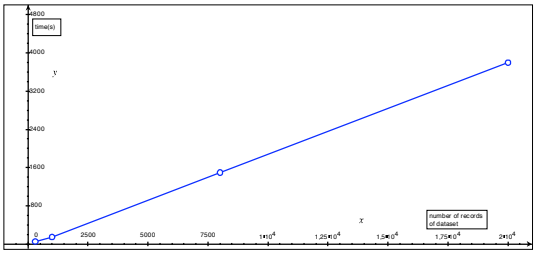


Figure 5: Execution time vs. dimensions of input datasets

acceptable (but can be refined by appropriate tuning some parameters). Algorithm EARL 2 exhibits very good performance but threshold values must be fixed for the matching attributes.

Finally, we ran an experiment with the objective to test the time efficiency of the two algorithms (EARL 1 and EARL 2 of course have the same time performance, implementing the same computational steps). In Figure 5, we show the time necessary for the execution versus the dimensions of the input datasets, to which we added a 20000 record datasets, obtained synthetically from the 8000 record one. We can observe that the such performance are really good: for the maximum size datasets, that is 20000 records each, the time is of slightly more than one hour.

6. RELATED WORK

In this section, we briefly review some works of the area of multiobjective optimization through evolutionary algorithms.

In 1985 Schafer [20] implemented VEGA (Vector Evaluated Genetic Algorithm), the first multiobjective genetic algorithm to find a non-dominate solution set. On the one hand this algorithm has the advantage of an easy implementation, on the other the algorithm could, with an high probability, prefer in the selection step solutions that are near the optimum only for one single objective.

In 1993 Hajela and Lin [18] proposed a genetic algorithm based on weights (WBGA). Each objective function is multiplied by a certain weight; the sum of these weighted objective functions represents the fitness function. The algorithm has a low complexity, but it is difficult to obtain a uniform diversity of non-dominated solutions, because a uniform diversity of the weights doesn't necessarily correspond to a uniform diversity of solutions.

Fonseca and Fleming in 1993 [11] introduced a multiobjective genetic algorithm (MOGA) that uses non-domination population classification. At each individual of the population a rank is assigned. This rank is calculated as a function of the number of individuals that dominate it. In this way non-dominated individuals have the same rank. In particular cases, the algorithm can produce an unexpected bias in the direction of some solutions in the search space.

In 1989 Goldberg proposed to use the concept of non-dominated sorting for genetic algorithms. This idea was implemented by Srinivas and Deb in 1994 [21] in Non-dominated Sorting Genetic Algorithm (NSGA). The algorithm performance is strongly dependent on the minimum distance that two solutions must have in order that one doesn't influence the other's fitness value.

In 1998 Zitler and Thiele [25] proposed an elitist evolutionary algorithm (SPEA). In each iteration the external population is populated with the best individuals. In case that the insertion of new individuals makes the external population exceed a certain dimension, it will be reduced with clustering techniques. Fitness assignment procedure is easy to calculate and a clustering algorithm ensures a good spread but a large external population increases the selection pressure for the elites and a small external population cancels the effect of elitism. Moreover there is a bias in fitness assignment dependent on the exact population and densities of solutions in the search space.

In 2000 Knowles and Corne [15] proposed Pareto-Archived Evolution Strategy (PAES). The approach consists in keeping a small/limited archive of non-dominated solutions. The archive is updated in the next generations in order to keep non-dominated solutions with a good degree of diversity. The problem of this algorithm is the definition of the archive's size: if too small it can loose optimal solutions, if too big it can make the algorithm inefficient.

The algorithm used in this paper, namely NSGA-II [5, 6], implements a specific strategy in order to keep the best individuals, and a mechanism for preserving the diversity among the different solutions. We have chosen this algorithm as a basis for EARL because it allows to have a Pareto ordering of the individuals in the population, without creating a unique, artificial macro-objective as a linear combination of the single objectives. This feature fits perfectly with the nature of our problem. Moreover, with respect to other algorithms similarly using Pareto-based ordering, a better efficiency is obtained by NSGA-II .

As far as we know, genetic algorithms haven't been used expressly for the RL process, but some of the approaches proposed for database merging provided a useful contribution to such a usage. In [9] a genetic algorithm is used for

the constrained statistical matching[3]. In this work the constrained statistical matching is reduced to the balanced linear transportation problem. The function to be minimized is the sum of the squares of the distances between the fields of the matched records. This approach is similar to WBGA where each objective has equal weights. For this reason it's hard to know between two solutions with equal fitness which solution is better in which objective. Zhu and Ungar [24] propose a technique for database merging that uses the dynamic programming to compute the distance between a couple of strings. This approach has a good accuracy after 150 generations, but it needs to define, for each domain, a dictionary in order to standardize data, otherwise the accuracy decreases significantly.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented an original formulation of the record linkage problem as a multiobjective optimization problem, and we have proposed an evolutionary algorithm, EARL, for solving it. The context in which we see the major application of this algorithm is the on-the-fly data integration one. In its current implementation, EARL has the following advantages if used in this context:

1. It is a *one-shot* algorithm. Specifically, record linkage is typically implemented as a process with different algorithms for each of its phases, such as a search space reduction phase which involves sorting algorithms, a phase where the appropriate comparison functions must be chosen, a decision phase with related algorithms, etc. Instead, EARL is a single algorithm that can be easily run even in complex contexts like the on-the-fly integration ones.

2. Being based on the evolutionary paradigm, it depends just on data, that is, EARL automatically adapts to data at hand with limited human knowledge required.

On-the-fly data integration is often supported by a peer-to-peer network where sources can be dynamically integrated as new peers joining an already built peer network. EARL would prove very effective if used in a peer-to-peer system, however, it still needs an accurate design to such a scope. We plan to address the issues regarding such design in the future. These issues include:

- the integration of the algorithm within the query answering process;
- the design of distributed mechanisms for enacting the algorithm in a peer to peer context;
- the physical distributed implementation of the algorithm.

Finally, we plan to perform extensive experimentation in order to test the behavior of the algorithm with other large size real datasets.

[3]Statistical matching is a procedure used to link two files or datasets where each record from one of the files is matched with a record from the second file that generally does not represent the same unit, but does represent a similar unit.

8. REFERENCES

[1] Thomas Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, Oxford, UK, 1996.

[2] C. Batini and M.Scannapieco, *Data quality: Concepts, methods and techniques*, Series on Advanced Database Applications, Springer, 2006.

[3] Gary B. Lamont Carlos A. Coello Coello and David A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, Springer, Norwell, MA, USA, 2007.

[4] K. Deb, *Multi-objective optimization using evolutionary algorithms*, Wiley-Interscience Series in Systems and Optimization, John Wiley & Sons, Chichester, 2001.

[5] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan, *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*, Proceedings of the Parallel Problem Solving from Nature VI Conference (Paris, France) (Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, eds.), Springer. Lecture Notes in Computer Science No. 1917, 2000, pp. 849–858.

[6] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: Nsga-ii*, IEEE Trans. Evolutionary Computation **6** (2002), no. 2, 182–197.

[7] A.K. Elmagarmid, G.I. Panagiotis, and S.V. Verykios, *Duplicate record detection: A survey*, IEEE Transaction on Knowledge and Data Engineering **19** (2007), no. 1.

[8] Ivan P. Fellegi and Alan B. Sunter, *A theory for record linkage*, Journal of the American Statistical Association **64** (1969), no. 328, 1183–1210.

[9] Giovanni A. Flores and Eliezer A. Albacea, *A genetic algorithm for constrained statistical matching*, (2007).

[10] D.B. Fogel, *Evolutionary computation. towards a new philosophy of machine intelligence*, The Institute of Electrical and Electronic Engineers, New York, 1995.

[11] Carlos M. Fonseca and Peter J. Fleming, *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*, Genetic Algorithms: Proceedings of the Fifth International Conference, Morgan Kaufmann, 1993, pp. 416–423.

[12] M.A. Jaro, *Advances in record linkage methodologies as applied to matching the 1985 census of tampa,florida*, JASS-84, 1985.

[13] J.H.Holland, *Adaptation in natural and artificial systems*, The MIT Press, Cambridge, 1992.

[14] jmetal, http://mallba10.lcc.uma.es/wiki/index.php/JMetal.

[15] Joshua D. Knowles and David W. Corne, *Approximating the nondominated front using the pareto archived evolution strategy*, Evol. Comput. **8** (2000), no. 2, 149–172.

[16] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava, *Record linkage: similarity measures and algorithms*, SIGMOD Conference, 2006, pp. 802–803.

[17] Wai Lup Low, Mong Li Lee, and Tok Wang Ling, *A*

knowledge-based approach for duplicate elimination in data cleaning, Inf. Syst. **26** (2001), no. 8, 585–606.

[18] P.Hajela and C.Y.Lin, *Genetic search strategies in multi-criterion optimal design*, Structural Optimization, 1992.

[19] Sam's String Metrics, http://www.dcs.shef.ac.uk/ sam/stringmetrics.html.

[20] J. David Schaffer, *Multiple objective optimization with vector evaluated genetic algorithms*, Proceedings of the 1st International Conference on Genetic Algorithms (Mahwah, NJ, USA), Lawrence Erlbaum Associates, Inc., 1985, pp. 93–100.

[21] N. Srinivas and Kalyanmoy Deb, *Multiobjective optimization using nondominated sorting in genetic algorithms*, Evolutionary Computation **2** (1994), no. 3, 221–248.

[22] W.E. Winkler, *Using the EM Algorithm for Weight Computation in the Fellegi and Sunter Modelo of Record Linkage*, Proceedings of the Section on Survey Research Methods, American Statistical Association, 1988.

[23] W.E. Yancey, *Improving em algorithm estimates for record linkage parameters*, RESEARCH REPORT SERIES, Statistics no.2004-01.

[24] J. J. Zhu and L. H. Ungar, *String edit analysis for merging databases*, Proceedings of the KDD-2000 Workshop on Text Mining, 2000.

[25] Eckart Zitzler and Lothar Thiele, *An evolutionary algorithm for multiobjective optimization: The strength pareto approach*, Tech. Report 43, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 1998.

The Role of Implicit Schema Constructs in Data Quality

Anthony Cleve
PReCISE Center/LIBD
University of Namur
21 rue Grandgagnage
B-5000 Namur, Belgium
acl@info.fundp.ac.be

Jonathan Lemaitre
PReCISE Center/LIBD
University of Namur
21 rue Grandgagnage
B-5000 Namur, Belgium
jle@info.fundp.ac.be

Jean-Luc Hainaut
PReCISE Center/LIBD
University of Namur
21 rue Grandgagnage
B-5000 Namur, Belgium
jlh@info.fundp.ac.be

Christophe Mouchet
ReveR S.A.
130 Boulevard Tirou
6000 Charleroi, Belgium
cmo@rever.eu

Jean Henrard
ReveR S.A.
130 Boulevard Tirou
6000 Charleroi, Belgium
jhe@rever.eu

ABSTRACT

This paper presents a comprehensive approach to legacy data quality assessment and improvement. It is based on an initial database reverse engineering phase that allows to recover *implicit* constructs, that is, structures and constraints that have not been explicitly declared in the database schema, using program analysis techniques. These constructs then serve as a basis for detecting data inconsistencies, identifying unsafe program fragments, and proposing necessary improvements at both the database and program sides.

1. INTRODUCTION

Information Quality (IQ) is now considered as an important research topic as much for practitioners explaining data as for research groups considering it as an important challenge. For industry IQ mainly has an economical impact underlined by authors through financial cost [25]. As a research challenge many definitions and methods have been proposed so far. Some of them may be considered as strongly accepted as those given by Ballou and Pazer [1], Wang and Wand [26] or Wang and Strong [27] that are the basis of the current definitions of data quality. Generally accepted terms for data quality include *accuracy*, *completeness*, *consistency* and *currency* [18]. In this context many researchers have proposed and still propose new approaches to:

- detecting erroneous data in a database;
- consistently querying a database containing wrong data;
- cleaning a database, by correcting or discarding data errors.

These works have been conducted in various research contexts like database integration, database migration, data warehouse feeding or reverse engineering. The same holds for the technological space that comprises traditional relational databases, object-relational databases, XML data repository, legacy systems (e.g., CODASYL DBMSs and IMS), standard files, etc.

Related works mainly concentrate on the definition of data quality and on the data cleaning process. According to the existing quality terminology this work addresses the data consistency problem. The latter generally reflects situations where different values of the same concept are present in the database, or where some data instances violate particular rules. In the 90's Wang and Wand [26] described data consistency in the following terms: *"a data value can only be expected to be the same for the same situation"*. More recently Scannapieco et al. [24] proposed a more precise definition: *"The consistency dimension captures the violation of semantic rules defined over (a set of) data items."* With *reference to the relational theory, integrity constraints are an instantiation of such semantic rules"*. Using a semiotic approach, Price and Shanks [20] used the concept of *"conforming to rules"* instead of the term *"consistency"* in their framework. The concept represents the fact that *"the data obeys business and other integrity rules"*. A lot of methods have been proposed in the purpose of detecting and resolving data defects [21]. The related literature comprises dedicated frameworks [11, 6, 18, 17], particular methods using data properties [10], statistical analysis techniques [7] as well as solutions based on heuristics [5].

While recent results on data quality mainly take the data themselves as inputs for detecting integrity problems, the contribution of our work is to also consider both the underlying database schema and the application programs. Moreover, we concentrate on the *intended schema*, that is, the physical schema as it is declared in the DDL code, augmented with *implicit constructs* recovered through reverse engineering techniques. Analyzing the data access behavior of the programs allows us to identify integrity constraints that are not explicitly declared in the DDL code. These additional constraints then serve as a basis for (1) assessing the quality of the data, (2) identifying unsafe critical code fragments, and (3) proposing adequate system improvements. The principles of the approach are illustrated in the context of legacy systems developed on top of relational databases. A real-world example of *CODASYL-to-DB2 migration* will then be discussed.

This paper is structured as follows: Section 2 describes in more detail the technical context of the inconsistencies. Section 3 presents a proposal for detecting implicit constraints and associated data inconsistencies. Section 4 elaborates on possible solutions for improving data filtering at both

database and program sides. In Section 5 we report on a tool-supported industrial experience for which we used the approach described in this paper. Our concluding remarks and research perspectives are presented in Section 6.

2. PROBLEM DESCRIPTION

As explained above, we focus the discussion on a particular kind of consistency problem: conflicting data against schema constructs (that is, structures and/or constraints). In this section, we further describe the problematic situation motivating our work as well as the concepts we will use in this paper.

2.1 The Problem of Data Validation

The formal rules defining data consistency may be associated to data structures or integrity constraints of various categories. Among them, let us mention:

- basic integrity constraints: primary keys, functional dependencies, cardinalities, types;
- referential constraints: foreign keys, inclusion constraint;
- existence constraints: exclusive (A and B cannot be not null at the same time), coexistence (A and B must be null or not null at the same time), implication (if B is not null, then A must not null either;
- domain constraints: interval of values, null values;
- derived values and intended redundancies.

We distinguish two main approaches to managing integrity constraints, namely database (DB) side data validation (Section 2.2) and program side data validation (Section 2.3).

2.2 Database Side Validation

According to the database side validation approach, the integrity constraints are managed by the database management system. Database side constraints may be of two forms. First, they can be declared by means of *native constructs* in the DDL code. Typical examples are primary keys, unique predicates , foreign keys, data type, not null columns and format constraints. The second category of constraints are specified by means of programmed SQL components such as check predicates, triggers and stored procedures.

The correct use of both validation solutions greatly reduces the risk of introducing inconsistencies in the database. Indeed, whatever the path used for modifying the data (direct access, programs, middleware, wrappers, ...) the specified constraints will always be verified by the DBMS, that will reject any modification request that attempts to violate constraints.

2.3 Program Side Validation

Integrity constraints validation may also be implemented within application programs. Different policies exist depending on the level of DB access centralization of the underlying system architecture:

- *Centralized management* relies on the use of data access component(s). Such a component, also known as *data wrapper* [19], provides the application programs with a database manipulation API (reading, insertion, modification, ...) that is also in charge of data consistency management, which consists in verifying that each requested database operation does not violate corresponding constraints.
- *Distributed management* captures the situation according to which data validation is scattered over the various application programs. Each program manipulating the database should verify the constraints that its behavior could potentially violate, which correspond to constraints that are not checked at the DB side. This pattern outlines the work of literature as far as integrity control is concerned, and, quite naturally, is the most popular.

We distinguish two main data validation strategies at the program side: *reactive validation* and *proactive validation*.

- *Reactive validation* verification tasks are performed before executing a database modification that could challenge data consistency. For instance, before inserting a new order the program verifies that the corresponding customer already exist in the database.
- *Proactive validation* consists in enforcing data integrity rules during the query construction process itself, in such a way that the executed database operations never violate integrity constraints. This kind of behavior is typically implemented using user interface restrictions. For example, when encoding a new room reservation the user must select the room identifier among the list of available room references.

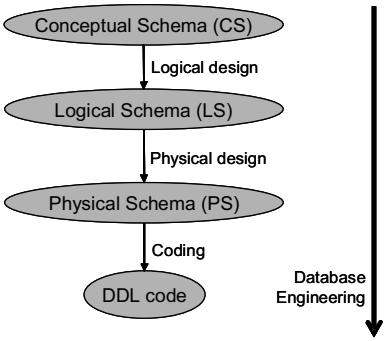


Figure 1: Standard database engineering process.

2.4 Database Engineering in an Ideal World

The process of designing and implementing a database that has to meet specific user requirements has been described extensively in the literature [3] and has been available for several decades in standard methodologies and CASE tools. It is made up of four main subprocesses, each producing a database schema at different levels of abstraction:

(1) *Conceptual design* is intended to translate user requirements into a conceptual schema, which is a platform-independent abstract specification of the future database. This schema collects all the information structures and constraints of interest.

(2) *Logical design*, which produces an operational logical schema that translates the constructs of the conceptual

schema according to a specific technology (entity without loss of semantics. The transformational approach to database engineering ([13]) allows this process to be automated.

(3) *Physical design*, which augments the logical schema with performance-oriented constructs and parameters, such as indexes, buffer management strategies or log management policies.

(4) *Coding*, which translates the physical schema (and some other artifacts) into the DDL (*Data Definition Language*) code of the database management system. Structural DDL declaration code SQL as well as SQL components such as checks, triggers and stored procedures are generated to code the information structures and constraints of the physical schema.

Figure 1 illustrates this ideal view of database engineering, according to which each construct expressed in the conceptual schema is (correctly) implemented into SQL objects. Let $C(s)$ denote the set of constructs expressed in a schema s. Then an ideal DDL code would be such that $C(DDL) = C(CS)$.

2.5 The Concept of Implicit Constructs

Unfortunately, many databases have not been developed in a disciplined way, that is, from a preliminary conceptual schema. This was true for old systems, but loose empirical design approaches keep being widespread for modern databases due, notably, to time constraints, poor database education and the increasing use of object-oriented middleware that tend to consider the database as the mere implementation of persistent classes. Secondly, the logical (DBMS-dependent) schema, that is supposed to be derived from the conceptual schema and to translate all its semantics, generally misses several conceptual constructs. This is due to several reasons, among which the poor expressive power of DBMS models, the lazziness, awkwardness or illiteracy of some programmers [4], or the need for performance.

From all this, it results that the logical schema often is incomplete and that the DDL code that expresses the DBMS schema in physical constructs ignores important structures and properties of the data. The missing constructs are called *implicit constructs* (*ImpC*), in contrast with the *explicit constructs* (*ExpC*) that are declared in the DDL of the DBMS.

Several field experiments and projects have shown that as much as half of the semantics of the data structures are implicit [15]. Therefore, merely parsing the DDL code of the database, or, equivalently, extracting the physical schema from the system tables, sometimes provides barely half the actual data structures and integrity constraints of the database. Hence the need for database reverse engineering [12], the goal of which is to recover the complete conceptual schema of a legacy database. In particular, it aims at identifying implicit constructs by analyzing programs source code, database procedural components, as well as other artefacts like screens, reports, or programmer guides (see Figure 2).

Among the set of *ImpC* we identify two categories based on the component(s) in charge of managing the constructs:

- $ImpC_D$ is the set of implicit constructs that are verified at the database side;

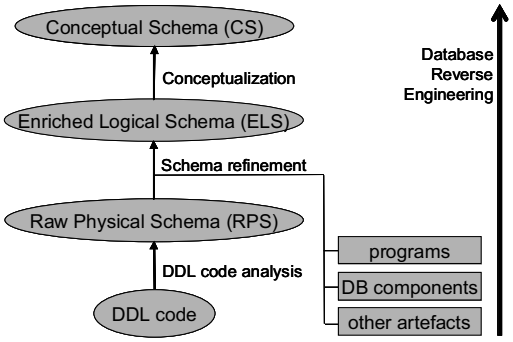


Figure 2: Implicit constructs recovery during data reverse engineering.

- $ImpC_P$ is the set of implicit constructs that are checked at the program side.

We can now define the set of implicit constructs as the difference between the conceptual schema and the DDL code: $ImpC = C(CS) \setminus C(DDL)$. Ideally, it should hold that $ImpC = ImpC_D \cup ImpC_P$, which means that each implicit construct is managed somewhere in the system. Note that it could happen that some constructs are validated at both the database and program sides, which translates as $ImpC_D \cap ImpC_P \neq \emptyset$.

3. ERROR DETECTION

The objective of the error detection phase is to identify data errors due to the violation of implicit constructs. This phase involves two successive problems:

1. recovering implicit constructs, and more specifically $ImpC_P$;
2. detecting data inconsistencies against those constructs.

3.1 Retrieving Implicit Constructs

The implicit constructs recovery process takes as main input the source code of the application programs. Our approach relies on the use of data-flow analysis techniques [8] applied to the source code, with a focus on database operations. The ideal goal of the analysis is to observe the way potential implicit constructs are *used* or *checked* by the application programs.

Our data flow analysis approach is based on the concept of *System Dependency Graph* (SDG), introduced by Horwitz and al.[16]. The SDG for program P is a directed graph whose nodes are connected by several kinds of arcs. The nodes represent assignment statements, control predicates, procedure calls and parameters passed to and from procedures (on the calling side and in the called procedure).

The arcs translates dependencies among program components. An arc represents either a *control dependency* or a *data dependency*. A control dependency arc from node v_1 to node v_2 means that, during execution, v_2 can be executed/evaluated only if v_1 has been executed/evaluated[1]. Intuitively, a data dependency arc from node v_1 to node

[1]The definition is slightly different for calling arcs, but this does not change the principle.

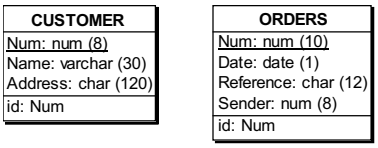


Figure 3: Two tables including implicit constructs

```
select substring(Address from 61 for 30), Reference
into   :CITY, :PRODUCT
from   CUSTOMER C, ORDERS O
where  C.Num = O.Sender
and    O.Num = :ORDID
```

Figure 4: Extracting hidden City and Product information (predicative join)

v_2 means that the state of objects used in v_2 can be defined/changed by the evaluation of v_1.

In order to illustrate the need of data flow analysis for database reverse engineering, let us consider the small but representative example based on the schema of Figure 3. This schema is made up of two tables describing customers and orders. It translates the constructs declared in the DDL code. The SQL query of Figure 4 obviously extracts the customer city and the ordered product keys of the order. It asks the DBMS to extract these data from the row built by joining tables `CUSTOMER` and `ORDERS` for that order. It exhibits the main features of the program/query interface: the program transmits an input value through host variable `ORDID` and the query transmits result values in host variables `CITY` and `PRODUCT`. The analysis of this query brings to light some important hidden information:

1. The join is performed on columns `Num` and `Sender`. The former is the primary key (the main identifying column) of table `CUSTOMER` while the second one plays no role so far. Now, we know that most joins found in application programs are based on the matching of a foreign key and a primary key. As a consequence, this source code fragment strongly suggests that column `Sender` is a foreign key to table `CUSTOMER`. Further analysis will confirm or reject this hypothesis.

2. The seemingly atomic column `Address` appears to actually be a compound field, since its substring at positions 61 to 90 is extracted and stored into a variable named `CITY`.

Translating this new knowledge in the original schema leads to the more precise schema of Figure 5.

Obviously, the implicit constructs recovered by program analyzers are only *candidate* integrity rules, that still have to be validated. This can be done via user validation or data analysis. Figure 6 depicts diagnosis box of the dependency validation tool we have developed as a plug-in of the DB-MAIN CASE tool, which allows the user to visualise the set of database fields dependencies detected by each analysis. For each resulting dependency, the tool provides the user with (1) the source and target database fields or columns, (2) their corresponding record types or tables, (3) a source code fragment (also known as *program slice*) from which the data flow has been derived, and (4) the corresponding source code location (program name and lines of code). Based on these results, the user may validate or discard the detected dependencies between fields. He can qualify each dependency as *no dependency*, *foreign key* or *redundancy* (5).

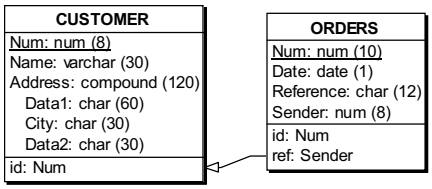


Figure 5: Potential implicit constructs revealed by source code analysis

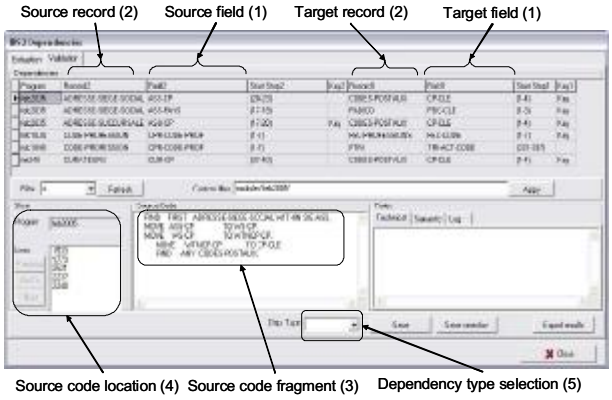


Figure 6: Result of data dependency analysis to be validated

3.2 Detecting Data Errors

Once the implicit constraints have been validated they can serve as a basis for assessing the quality of the data themselves, by detecting inconsistent data against recovered constraints. The approach we have developed consists in automatically generating database queries looking for violation of the implicit constraints specified in the enriched logical schema. As an example, the following query will be produced from the implicit referential constraint of Figure 5:

```
select *
from ORDERS
where Sender not in (select Num
                     from CUSTOMER)
```

This query retrieves all the instances of table `ORDERS` that do not reference an existing row of table `CUSTOMER`.

4. QUALITY IMPROVEMENT

The goal of the system improvement phase is twofold. Firstly, the database must be made consistent with respect to the implicit constructs that have been recovered. This data cleansing process involves the correction/deletion/isolation of the erroneous data identified during the previous step. Secondly, we have to make sure that such data inconsistencies will to occur in the future. This necessitates to check that each implicit constraint is correctly and fully managed either at the database side or at the program side.

4.1 Data Correction

Data correction approaches can be divided into three categories [11]: manual, semi-automatic and automatic. Manual correction is generally the least cost-effective solution but is the more accurate (using competent people). Automatic correction requires a lower cost but relies on predefined strategy based on rules and heuristics. So the accuracy may decline depending of the context. Semi-automatic solutions represent a balanced compromise between manual and fully-automated in terms of cost and accuracy. Semi-automatic data correction seems to be the most frequent scenario. A threshold may be defined so that when a confidence value fall under the threshold then the values have to be fixed manually. Otherwise the correction is done automatically. Another example is the knowledge-based framework developed by Low et al. [17] working as an expert system for data correction. Finally, the interactive system proposed by Raman and Hellerstein [22] allows the user to gradually build the correction plans.

In the present work, data correction itself is rather seen as a business problem. Indeed, although automated analysis may allow to detect data inconsistency problems, fixing these problems often remains under the responsability of the business. For instance, let us suppose that data analysis detects a set of `ORDERS` with an invalid value of `Sender`. From a technical point of view, several correction strategies are possible: discarding the orders, correcting the orders, introducing new customers, or relaxing the referential constraint. But choosing between these solutions is rather a business decision.

4.2 Improving DB side Data Validation

Improving data integrity at the DB side can be done by introducing additional rules corresponding to the recovered implicit constraints. Such rules can be expressed within various kinds of components like new native constraints (unique, foreign key), checks, triggers or stored procedures.

Although most of the additional code fragments can be automatically generated, in practice their development may also necessitate human decisions. This holds, for instance, for the delete/update mode selection in the case of referential constraints (*cascade*, *no-action*, *set default* or *set null*).

Some data filtering improvements may necessitate the migration of the data instances involving a possibly complex ETL process. However, other improvements may have a negative impact on the successful execution of existing programs. In some situations, the use of a fault-tolerant filtering strategy may be preferable. According to such a strategy data errors may still be introduced but the filtering component systematically reports on those errors. Error reports then serve as a basis for (1) correcting erroneous data instances, (2) identifying the sources of errors in the application programs or the underlying business processes and (3) proposing necessary improvements. This idea of temporarily tolerating inconsistency has been proposed by Balzer [2].

4.3 Improving Program side Validation

The second improvement strategy concerns the application programs and particularly focuses on their data validation code fragments. It aims at identifying and fixing the unsafe data access paths that could (have) lead to the insertion of inconsistent data with respect to implicit constraints.

In theory, the set of implicit constraints that are considered during program side validation are the constraints that are not managed at the DB side (after DB side filtering improvement). In practice, some constraints that are already managed by the DBMS may also be considered in order to verify that they are not also checked at the program side. Such double validations should be avoided for obvious performance reasons.

4.3.1 Identifying Unsafe Data Access Paths

The identification of unsafe access paths is quite similar to the implicit constraints recovery process. However, it slightly differs from the latter in that it focuses on a subset of database operations accessing a subset of database entity types. Indeed, its goal is to analyze *critical database operations*. A critical database operation with respect to a constraint c is a database modification instruction that could potentially violate c. An *unsafe data access path* with respect to an implicit constraint c is a source code fragment that includes at least one critical database operation for c and that does not (fully) prevent c to be violated.

A typical example is the implicit referential constraint. Let us consider again the implicit foreign key Sender (denoted by R) from table `ORDERS` to table `CUSTOMER`. The following database operations are critical for R:

- `update CUSTOMER`, where the primary key value is modified;
- `delete from CUSTOMER`;
- `insert into ORDERS`;
- `update ORDERS`, where the foreign key value is modified.

Each program executing one of these operations must make sure to preserve data consistency against R. This verification should be done via a validation query selecting rows from table `CUSTOMER` (resp. `ORDERS`) for operations on table `ORDERS` (resp. `CUSTOMER`). In that particular context, detecting unsafe data access paths could be done by looking for source code locations where critical operations described above are not preceded by a suitable verification query, or that do not behave correctly according to the result of such a query.

Figure 7 gives an example pseudo-code fragment involving a correctly validated critical operation. Before inserting a new row in table `ORDERS`, the program checks that the value of variable `NewSender` corresponds to an existing customer reference. This example corresponds to a typical code pattern of *reactive* program-side validation. According to this code pattern, the dependency between the critical operation and the verification query is twofold. First, the critical operation will be executed if and only if the result of the verification query is satisfying. Second, the input arguments of the verification query are included as input arguments of the modification query. In other words, there exists a data dependency relationship between the inputs variables of the validation query and a subset of the input variables of the critical operation (variable `NewSender` in our example).

In the case of proactive validation of an implicit referential constraint, the inter-query dependency is of another nature. Indeed, the objective of the validation query is rather to retrieve the set of valid foreign key values (i.e., the set of existing primary keys in the target table). The user must then select one value among this set. The insert operation

```
exec sql
  select count(*) into :NBR
  from CUSTOMER
  where CustNum = :NewSender
end-exec
if (NBR = 0)
   display('unknown customer - insertion aborted !')
else
   exec sql
     insert into ORDERS(Num, Date, Reference, Sender)
     values(:NewNum, :NewDate, :NewRef, :NewSender)
   end-exec
end-if
```

Figure 7: Sample code fragment with implicit referential constraint validation

then uses this value as input parameter. Thus, in this case, there exist a data flow relationship between the output variables of the validation query and the input variables of the critical operation.

4.3.2 *Fixing Unsafe Data Access Paths*

When unsafe data critical operations are detected, different solutions can be considered. The most obvious correction consists of direct code insertion. The program source code is adapted such that each critical operation requires a corresponding validation step.

There typically exists several solutions. In our example of implicit foreign key, an unsafe delete statement on table `CUSTOMER` can be transformed in different ways. First, we can add a verification query, that checks if the customer to discard still has orders in the database. If this is the case, the deletion is aborted (i.e., no action mode). Another adaptation could be to systematically delete all associated `ORDER` rows before deleting each `CUSTOMER` row (i.e., cascade mode). In both cases, the program adaptation can be automated.

Another program improvement strategy consists in reorganizing the data manipulation code fragments by introducing an additional layer, which provides the application programs with a database access API. This data access module, or *wrappers*, a large part of which can be generated, is in charge of managing data consistency when performing critical operations. The legacy program code can be interfaced with this new component with minimal adaptation. Such a source code refactoring allows to better modularize, and therefore to better master the data consistency management concern. This approach has been reported in [25].

5. INDUSTRIAL APPLICATION

The approach and the tools that support it have been recently used in the context of an industrial database migration project, conducted by ReVeR S.A. The goal of this project was to migrate a CODASYL database (IDS/II) towards a relational platform (DB2). The legacy system is in use in a Belgian federal administration. More information on our migration approach itself can be found in [14, 9].

The physical schema comprises 42 areas, 112 record types, 73 sets (relationship types), 1721 fields. A database reverse engineering phase allows us to recover the complete logical schema including implicit constraints (among which 14 redundant foreign keys, multivalued norms, and heterogeneous field decompositions (a total of 1511 fields among which 655 are optional).

The target DB2 schema include 147 tables, 1843 columns and a total of 144 foreign keys. A subset of the foreign keys have not been explicitly declared in the DDL code. They are rather managed by fault-tolerant triggers that produce a warning in a dedicated log table each time the relational constraint is violated by a database operation. These triggers were automatically generated. This approach allows to better master the referential constraints while avoiding to disturb the execution of legacy applications.

5.1 Quality Analysis Process

In the context of database migration, the problem of data quality analysis is of primary importance, a data cleaning process being often necessary. Indeed, the migrated data must respect the DDL constraints of the target database in order to be succesfully migrated. More generally, data migration appears as an excellent opportunity to perform some data cleaning as well as to apply some improvements at the program level.

During the data error detection phase, we particularly focused on format constraints (e.g., dates, zip codes, optional and default values) and implicit foreign keys. This phase involved the following steps:

1. The legacy IDS/II data were loaded into an intermediate DB2 database, in which each column is of type string, allowing error-free data migration. The DB2 database contained more than 415 millions of rows.
2. The underlying database schema was enriched with the following annotations:
   - The expected format of each DB2 column;
   - The validity constraints associated to each column (expressed as SQL *where* clauses);
   - The implicit foreign keys between tables.
3. Starting from this annotated schema, a data analysis program generated SQL queries for inspecting the intermediate DB2 database. This analysis produced an html document reporting on the detected data errors.

Figure 8 depicts an excerpt of this report. For each table, the report details the associated format constraints that are satisfied or violated by the data. For each violated constraint, it indicates the number of problematic instances and allows the user to browse the inconsistent records (via the 'Details' link). A separate part of the report is dedicated to foreign key inconsistencies. In this project, 76 implicit foreign keys have been considered, among which 24 proved to be violated by the legacy data. Regarding data format, a total of 3497 SQL queries were executed, allowing the detection of errors in almost 70% of the tables.

A typical problem was the presence of invalid dates. In the legacy database, date fields are represented as numeric values of the form 'YYYYMMDD'. In the target DB2 database, they are expressed as columns of SQL type Date. We encountered a large amount of inconsistent dates like '20070931'. This is mainly due to the behavior of some application programs considering day '31' as the last day of the current month, whatever the month. We also encountered special date values like '00000000' or '99999999' that are used by programs for simulating the *null* value and a future date, respectively. In total, around 35 millions of invalid date values were found.

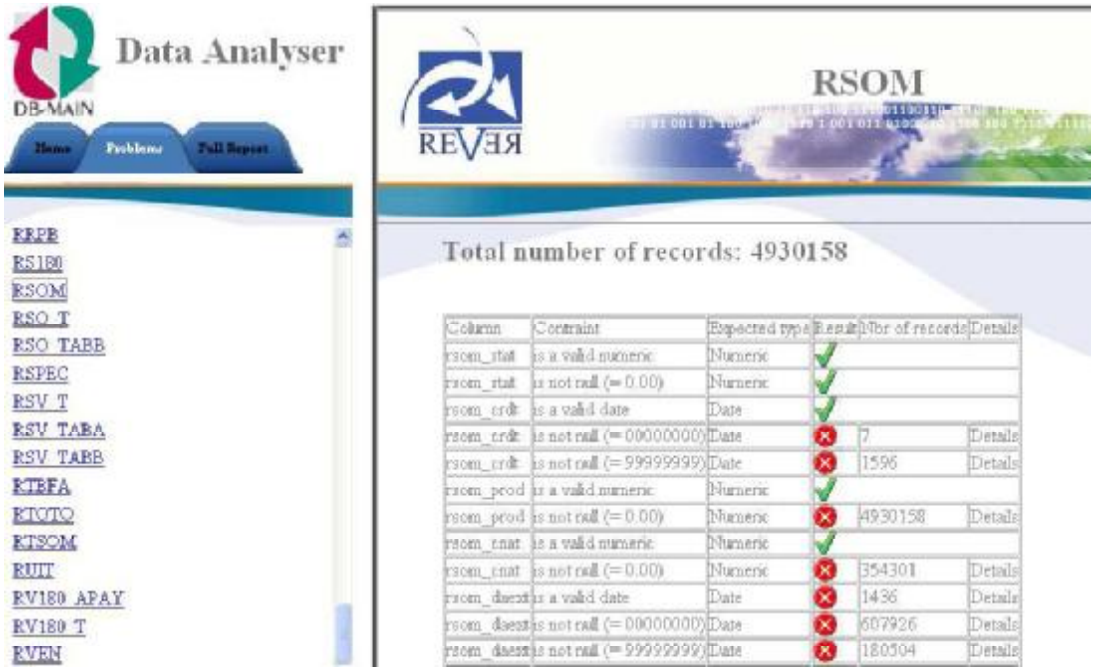


Figure 8: Excerpt of the error report produced by the data analyser

5.2 System Improvements

Based on the error report, the customer invested one complete week for (1) improving the quality of the data and (2) adapting the most problematic application programs. The latter improvements included the correction of the '*last day of the month*' problem, and the standardization of default values (e.g., a date '00000000' becomes *null*). As expected, a subset of erroneous data items could not be cleaned, which required to relax some constraints. For instance, a subset of fields representing invalid dates had to be migrated as numeric columns. Most of the (still) implicit constraints are now managed by dedicated triggers introduced at the DB side. These triggers systematically produce warnings when new data inconsistencies are inserted in the database.

5.3 Lessons Learned

This industrial project showed us that completely cleaning the entire database was not achievable, especially because the migrated data actually contains legal information that cannot be modified. Moreover, a subset of the data originates from external administrative entities which complicates data correction. In such situations, the only realistic solution consists in tolerating inconsistencies by temporarily relaxing some constraints.

The IT management people at the customer side did react very positively to our results. They spontaneously decided to invest a significant effort in the correction of most data errors. They also intend to undertake further adjustments to the application programs in the next future.

Another important conclusion is that assessing the quality of constantly evolving data is more than challenging. Ideally the database should be analysed everyday, which is not feasible in practice. We also learned that while data analysis may serve as a basis for validating candidate integrity constraints, it can also contribute to the identification of new implicit constraints. In the case of this project, detecting special date values for a given column suggests that the latter should be declared optional.

6. CONCLUSIONS

The paper describes a framework for data evaluation, data correction and program improvement based on implicit construct violation. This data defect source is both very important and prone to automated processing. The comprehensive framework described in this paper encompasses the data error identification process, such as the identification of the source of the errors in application programs (the *unsafe data paths*) and the correction of the latter. Its main contributions are the complete treatment of this important source of errors thanks to now mature reverse engineering techniques [12], the use of program analysis and transformation techniques, and the improvement of the whole application system, covering data, schemas and programs. In addition, the framework is tolerant to practical constraints such as the preservation of programs whose execution relies on data violation. The approach is supported by a suite of analysis, reporting, generation and transformation tools and has been successfully applied on a large administrative system migration.

It is important to note that the framework is fully generic, and can be applied to any data source, whatever its underlying data model. It has been experimented on an IDS/II to DB2 migration, but most of its principles and tools are platform-independent and can be quickly customized to any technology. In particular, being based on the DB-MAIN

database engineering approaches and tools, it benefits from its generic meta-model and model-independent transformation technology [13].

Beyond load replacement and further validation through additional industrial projects, an important way of improvement of the approach consists in applying learning techniques to increase the automation part of the program correction process. Indeed, most program corrections obey to systematic parametrized patterns. Learning from these heuristics, an intelligent correction help system can be designed to anticipate the kind of modification that should be applied in each case. Such an approach has been used in data correction ([22]) and can be applied to program correction as well. We plan to work in this direction.

7. REFERENCES

[1] D. P. Ballou and H. L. Pazer. Modeling data and process quality in multi-input, multi-output information systems. *Management Science*, 31(2):150–162, feb 1985.

[2] R. Balzer. Tolerating inconsistency. In *Proc. of the 13th international conference on Software engineering (ICSE'91)*, pages 158–165, 1991.

[3] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual database design: an Entity-relationship approach*. Benjamin-Cummings Publishing Co., Inc., 1992.

[4] M. R. Blaha and W. J. Premerlani. Observed idiosyncracies of relational database designs. In *Proc. of the Second Working Conference on Reverse Engineering (WCRE'95)*, page 116, Washington, DC, USA, 1995. IEEE Computer Society.

[5] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In Özcan, editor, *SIGMOD Conference*, pages 143–154. ACM, 2005.

[6] M. Bovee, R. P. Srivastava, and B. Mak. A conceptual framework and belief-function approach to assessing overall information quality. *International Journal of Intelligent Systems*, 18(1):51–74, jan 2003.

[7] R. Bruni and A. Sassano. Errors detection and correction in large scale data collecting. In *IDA*, volume 2189 of *Lecture Notes in Computer Science*, pages 84–94. Springer, 2001.

[8] A. Cleve, J. Henrard, and J.-L. Hainaut. Data reverse engineering using system dependency graphs. pages 157–166, Washington, DC, USA, 2006. IEEE Computer Society.

[9] A. Cleve, J. Henrard, D. Roland, and J.-L. Hainaut. Wrapper-based system evolution - application to codasyl to relational migration. In *Proc. of the 12th European Conference in Software Maintenance and Reengineering (CSMR'08)*, pages 13–22. IEEE Computer Society, 2008.

[10] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07)*, pages 315–326. ACM, 2007.

[11] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. Technical Report RR-3742, INRIA, 1999.

[12] J.-L. Hainaut. Introduction to database reverse engineering. LIBD Publish., 2002.

[13] J.-L. Hainaut. The transformational approach to database engineering. In *Generative and Transformational Techniques in Software Engineering*, volume 4143 of *Lecture Notes in Computer Science*, pages 95–143. Springer, 2006.

[14] J.-L. Hainaut, A. Cleve, J. Henrard, and J.-M. Hick. Migration of legacy information systems. In T. Mens and S. Demeyer, editors, *Software Evolution*, pages 105–138. Springer, 2008.

[15] J. Henrard and J.-L. Hainaut. Data dependency elicitation in database reverse engineering. In *Proc. of the Fifth European Conference on Software Maintenance and Reengineering (CSMR'01)*, pages 11–19, Washington, DC, USA, 2001. IEEE Computer Society.

[16] S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. *ACM Transactions on Programming Languages and Systems*, 12(1):26–60, Jan. 1990.

[17] W. L. Low, M.-L. Lee, and T. W. Ling. A knowledge-based approach for duplicate elimination in data cleaning. *Inf. Syst.*, 26(8):585–606, dec 2001.

[18] D. Milano, M. Scannapieco, and T. Catarci. Using ontologies for xml data cleaning. In *OTM Workshops*, pages 562–571. Springer, 2005.

[19] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. Ullman. A query translation scheme for rapid implementation of wrappers. In *Proc. Int'l Conf. Declarative and Object-oriented Databases*, 1995.

[20] R. J. Price and G. C. Shanks Empirical refinement of a semiotic information quality framework. In *HICSS*. IEEE Computer Society, 2005.

[21] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[22] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *VLDB*, pages 381–390. Morgan Kaufmann, 2001.

[23] T. C. Redman. *Data Quality for the Information Age*. Artech House, Inc., Norwood, MA, USA, 1997. Foreword By-A. Blanton Godfrey.

[24] M. Scannapieco, P. Missier, and C. Batini. Data quality at a glance. *Datenbank-Spektrum*, 14:6–14, aug 2005.

[25] P. Thiran, G.-J. Houben, J.-L. Hainaut, and D. Benslimane. Updating legacy databases through wrappers: Data consistency management. In *Proc. of the 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 58–67. IEEE Computer Society, 2004.

[26] Y. Wand and R. Y. Wang. Anchoring data quality dimensions in ontological foundations. *Commun. ACM*, 39(11):86–95, nov 1996.

[27] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12(4):5–30, 1996.

On APIs for Probabilistic Databases

Lyublena Antova and Christoph Koch
{lantova,koch@cs.cornell.edu}
Cornell University

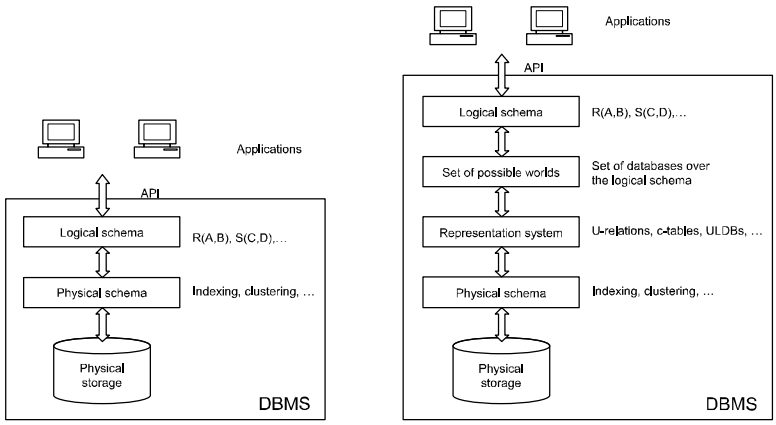
Abstract. We study database application programming interfaces for uncertain and probabilistic databases and present a programming model that is independent of representation details. Conceptually, we use the possible worlds semantics, and programs are independently evaluated in each world. We study a class of programs that appear to the user as if they are running in a single world rather than on a set of possible worlds. We present an algorithm for efficiently verifying this property. We discuss how updates can be implemented in uncertain database management systems, and propose techniques for optimizing database programs.

1 Introduction

In the last years there has been a profusion of research on managing uncertain and probabilistic data in different application scenarios such as Web information extraction, data cleaning, and tracking moving objects [1, 3, 4, 8–10]. Research on managing uncertainty has been focused on space-efficient models for representing uncertain information, query languages and efficient query processing, as well as confidence computation and ranking. Several systems for managing uncertain data are being developed (see e.g. [4, 3, 1]). However, no mature systems have evolved that support application development for uncertain databases.

A widely used approach to managing uncertain data is the possible worlds model. For example, in a moving object tracking scenario there can be several possible guesses for the identity of an object where the exact one is not known for sure. Each such option corresponds to one possible world. Under the possible worlds semantics, a query on an uncertain database is conceptually evaluated on each world independently, and the result is added to that world. In the special case when the possible world-set consists of a single world, this semantics coincides with the standard query evaluation semantics. In many scenarios however the set of possible worlds can be very large, or even infinite. Systems for managing uncertain data usually employ a compact representation for storing the possible worlds: e.g. MystiQ uses the tuple-independence model, Trio's representation is called ULDBs (databases with uncertainty and lineage), and MayBMS uses U-relations. All these systems rewrite queries on the set of possible worlds into queries on the representation.

An important component missing from current systems for managing uncertain data is the ability to write database application programs that access and update data through an application programming interface (API). Current



(a) Traditional DBMS (b) DBMS for uncertain data

Fig. 1. Levels of abstraction provided by a Database Management System.

systems either allow only batch execution of SQL-like queries on the uncertain database with no user interaction, or have a representation-dependent programming model that requires knowledge of system-specific implementation details.

Traditional database management systems abstract away the physical details of how the data is stored. External applications interact with the database by formulating queries and updates on the *logical schema*, which are then translated into operations on the physical storage structures on disk. Figure 1 shows an architecture of a traditional DBMS, and one of a DBMS for uncertain data. Compared to traditional DBMSs, systems for managing uncertain data need to deal with two additional levels of abstraction, the representation system and the possible worlds model.

To demonstrate some of the challenges of designing an API for uncertain databases, consider the following example (essentially from [3]). Figure 2 shows an example of a police database containing reports on stolen cars. Due to conflicting or missing information, several instances are possible, as shown in the figure. An application that manages such data should provide users with the ability to update or insert new evidence regarding observed objects, execute queries, or apply expert knowledge to resolve inconsistency.

The following program allows to enter new evidence about a stolen car. In case the car already exists in the database, and the information about it matches the user's input, the program increments the number of witnesses; otherwise a new entry is inserted into the database.

```
read("Enter license plate:", $x);
if (exists select * from cars where num=$x){ // modify existing entry
    for($t in select * from cars where num=$x){
      write("Current entry: $t");
      read("New location and color:", $loc,$color);
      if (exists select * from cars where num=$x and loc=$loc and color=$color)
```

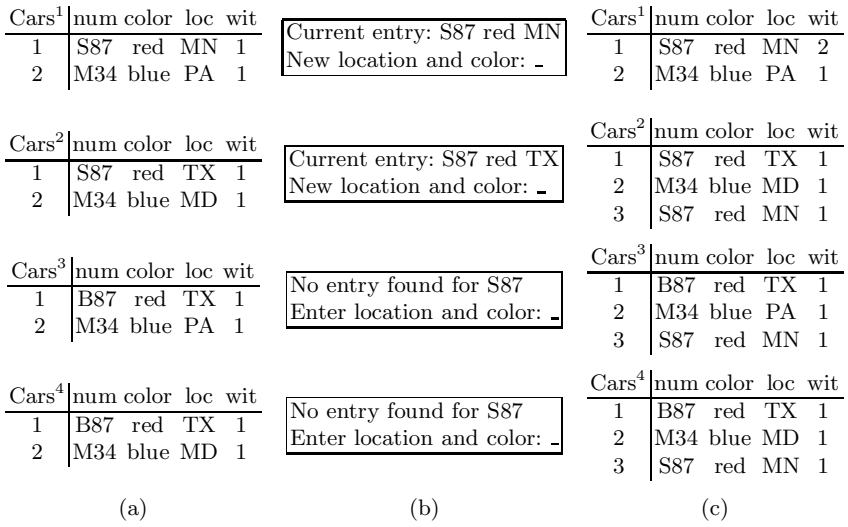


Fig. 2. Executing programs on uncertain databases: (a) set of possible worlds reporting information on stolen cars; (b) output of the program in each of the worlds of (a); (c) result of running the program on the world-set of (a).

```
        update cars set wit=wit+1 where num=$x and loc=$loc and color=$color;
      else
        insert into cars values($x,$loc,$color,1);
    }
}
else { // entry does not exist
    write("No entry found for $x");
    read("Enter location and color:", $loc, $color);
    insert into cars values($x,$loc,$color,1);
}
```

The program makes sense in the case of certain databases and is straightforward to execute. What is different in the presence of uncertainty? Consider the world-set in Figure 2 (a). The four worlds differ in the license plate number specified for car 1, and the reported location for cars 1 and 2. If the user enters S87 for the license plate number, in worlds 1 and 2 the car already exists in the database, so its entry will be shown to the user. In worlds 3 and 4 the car is not found and the user will receive a message reporting that. As shown in Figure 2 (b), there are three different outputs for the four worlds of Figure 2 (a), and the meaning of the expected input is also different. Figure 2 (c) shows the outcome of the program on each of the worlds if the user specifies S57, MN, red as input in each of the worlds. In world 1 the witness count for S57 is increased to 2, and in the remaining worlds a new tuple is added to the database.

Clearly, we cannot expect users to supply input for each world individually, or to deal with different output produced in each of the worlds. Such programs are not only unintuitive to the user: they are also infeasible to implement as in

practice the number of possible worlds can be prohibitively large. One solution to this problem is to introduce an intermediate level between the database and the user that verifies that the messages (output and input) that are passed between the database and the user are the same *in all worlds*, and collapses those into one. This has the advantage of hiding the uncertain nature of the data from the users, allowing them to work with the database as if it were complete. We say that in this case the program is observationally deterministic. A second approach is to only allow programs that are *guaranteed* to have the same behavior on all worlds, and to verify observational determinism in a *static manner*. For example, the above program can be modified to first request all evidence information, and then carry out the update in each of the worlds by either updating existing tuples or inserting new ones, without disclosing this different behavior to the user.

The second major challenge in API development for uncertain databases is to map programs that are conceptually executed on each world individually into programs on the representation. There have been studies on how to do this efficiently for queries. Programs are more complex as they provide richer structure such as updates, looping and branching constructs and the flow of execution can be different in each world. We take the approach of first pushing as much of the program code as possible into (set-at-a-time) queries and updates, which we then map to queries and updates on representations.

In this paper we present novel techniques for database programming. The contributions of this paper are:

- **Updates.** Updating uncertain databases presents a challenge as often a compressed representation is used that stores a single copy of a tuple appearing in multiple worlds. Updates can require decompressing the representation to allow changing a tuple in some worlds only. We discuss techniques for implementing updates on several recent representation systems, such as the U-relations of MayBMS [1]. Our techniques preserve compactness of the representation despite the need for decompression.
- **Programming model.** We describe a programming model for developing applications for uncertain databases where users can interactively execute queries and updates on the database and process the results in a high-level language. Our model is independent of the underlying representation. For that we adopt the possible worlds semantics. Conceptually, programs run on all worlds in parallel.
- **Observational determinism.** We study a class of programs whose behavior on uncertain databases is indistinguishable to the user from that on complete/certain databases, without restricting the way programs behave in the background where no user interaction occurs. We call such programs observationally deterministic and argue that this property is crucial if we want to design efficient programs for uncertain databases with intuitive user friendly interfaces. We also devise an efficient algorithm for deciding *statically* whether a program satisfies observational determinism. The underlying idea consists of examining the output sent to the user in terms of the query that produces it and checking whether the query can return uncertain results. As a side effect we obtain an efficient heuristic for deciding tuple q-certainty,

i.e. whether a tuple is certain in the answer to a query. The heuristic involves positive relational algebra operators only, which are often efficiently implementable on succinct representations.

– **Optimizing database programs.** Avoiding iteration over the possible worlds is crucial for achieving efficient execution. For that we need to map programs with update operations into ones that execute in bulk on all worlds. However, it is not clear how to deal with branching and for-loops in the programs, as the control flow can be very different in each world. For that we provide rewrite rules that map nested update programs into sequences of simple update programs that execute on all worlds. These results, while important in the context of uncertain databases, are relevant also in the case of certain databases.

2 Database Programming

A database programming model enables the development of applications that access and manipulate data stored in a database from a high-level programming language. A database program connects to a DBMS and can execute update commands, or issue queries and obtain cursors to iterate over the result. APIs provide means of accessing the data without knowing how the data is stored on disk, and often allow for porting programs between different database management systems. We next propose and study the properties of a programming model for uncertain data.

2.1 Queries and updates on uncertain DBMSs

We consider queries and updates specified using the SQL select, insert, update and delete statements. We take the possible worlds semantics to define the meaning of queries and updates. A query, applied on a set of possible worlds extends each world with the result of the query in that world. Similarly, an update operation is executed on each world of the world-set. For querying we will also consider the operator *conf* for computing the confidence of the possible tuples in the result of a query. The confidence of a tuple t is defined as the sum of the probabilities of all worlds containing t. We also consider two special cases of this operator: *possible*, which computes all possible tuples, and *certain*, which returns the tuples appearing in all possible worlds (see e.g. [2]).

There have been a number of studies on how to evaluate queries on uncertain databases by translating them into queries on the representation, see e.g. [6, 3, 1]. None of the present works however has considered the problem of applying updates on uncertain databases. As with querying, we would like to execute updates on the representation rather than iterate over the possible worlds.

Most systems for managing uncertain data use a compact representation for storing large sets of worlds. This usually means that tuple or attribute values that appear in several worlds are stored only once, with additional constraints that describe to which worlds they belong. Correlations are represented by means

of lineage in Trio, using world-set descriptors in MayBMS, and with graphical models in [9].

U_1	D_1	TID	A
	$(x_1 \rightharpoonup 1, 0.2)$	t_1	1
	$(x_1 \rightharpoonup 2, 0.8)$	t_1	2
	$(x_2 \rightharpoonup 1, 0.6)$	t_2	3
	$(x_2 \rightharpoonup 2, 0.4)$	t_2	4

U_2	D_1	TID	B
	$(y_1 \mapsto 1, 0.1)$	t_1	1
	$(y_1 \mapsto 2, 0.9)$	t_1	2
	$(y_2 \mapsto 1, 1)$	t_2	2

(a)

U_1	D_1	D_2	TID	A
	$(x_1 \rightharpoonup 1, 0.2)$	$(y_1 \mapsto 1, 0.1)$	t_1	8
	$(x_1 \rightharpoonup 1, 0.2)$	$(y_1 \mapsto 2, 0.9)$	t_1	1
	$(x_1 \rightharpoonup 2, 0.8)$	$(y_1 \mapsto 1, 0.1)$	t_1	8
	$(x_1 \rightharpoonup 2, 0.8)$	$(y_1 \mapsto 2, 0.9)$	t_1	2
	$(x_2 \rightharpoonup 1, 0.6)$		t_2	3
	$(x_2 \rightharpoonup 2, 0.4)$		t_2	4

(b)

Fig. 3. Updating uncertain databases: (a) U-relational database; (b) relation U_1 after applying the update of Example 1.

We will use as running example the U-relational database of Figure 3 (a) representing a relation $R(A, B)$ in an uncertain database. It consists of two vertical partitions $U_1(D_1, TID, A)$ and $U_2(D_1, TID, B)$ containing the possible values for the A and the B attribute of R, respectively. The column D_1 in the two relations is used to specify correlations of the possible values. For example $t_1.A$ has a value of 1 whenever the variable x_1 is mapped to 1 (which happens with probability 0.2), and with probability 0.8 has value 2 whenever $x_1 \rightharpoonup 2$. In this way the U-relational database represents compactly eight possible worlds, one for each of the possible combinations of values for the variables x_1, x_2, y_1, y_2.

Example 1. Let T1 be an operation that updates R:

```
T1: update R set A = 8 where B = 1;
```

This operation will update tuple t_1 for the worlds where $B = 1$. These worlds are constructed by taking y_1 to be 1. However, the current representation does not capture the desired correlation; therefore we need to create two copies of each of the alternatives of tuple t_1 in U_1 for the cases where it has to be updated or not. To compute the result we have to undo part of the decomposition, as shown in Figure 3 (b). ❙❙

Intuitively, each update operation consists of two steps: in the first one we compute copies of those tuples that will need to be updated in some worlds only, and in the second step we execute the update. The first step only changes the representation, but not the world-set itself. We only decompress when it is necessary – when in some world the attribute value needs to be updated, and we do not merge in tuples that will not be updated in any world.

We can think of such an update operation as consisting of two phases. In the first phase we split, or create copies of those tuples that need to be updated, and then in the second phase we update the values of those that satisfy the where-condition. In the first phase we only change the physical representation but the represented set of worlds remains the same.

construct	meaning
update statements	Execution of SQL create table, insert, update and delete queries. The query statements can be constructed using constants, values read from the database or user-supplied values.
read($\$x$,$\$x_{in}$);	Read user input into variable $\$x_{in}$. The user is displayed a prompt $\$x$.
write($x);	User output operation. The program can output messages to the user, including values stored in tuple variables.
+,-,*,/	Arithmetic operations on variables and constants.
for($t in Q){P}	Iterate over the result of a query Q and execute the nested program P for each binding of the tuple variable $t. The query language we consider is an extension of SQL with the keywords **possible** and **certain**, that compute the tuples appearing in some, or all worlds, respectively, and the construct **conf** returning the confidence of a tuple in the result of a query.

Table 1. Language constructs for database programming.

2.2 Programming interface

The programming language we will use is summarized in Table 1. As in the classical setting of certain databases, a database program is a sequence of statements that can execute select and update commands on the database, iterate over query results and provide user interaction through read and print commands.

How are programs executed on an uncertain database? We strive for a model for database programs for uncertain databases that satisfies the following desiderata. *First*, the execution model should be independent of the underlying uncertain database management system. This is important as it will allow porting of programs between different uncertain DBMSs. *Second*, programs should allow for efficient execution. *Third*, despite the fact that data is uncertain, programs should have intuitive user interfaces and should not expect users to be aware of the uncertainty.

To satisfy the first requirement, we naturally adopt the possible worlds semantics that has been the standard semantics used to define the meaning of queries on uncertain databases. According to this, the program is executed in all worlds in parallel; within a world it behaves in the same way as on a complete database; all updates the program makes are applied to the current world. Of course, a direct implementation of this semantics is unrealistic as there are far too many worlds that can be represented by an uncertain database. In the context of querying several works have studied [6, 1, 3] how to avoid iterating over the possible worlds and evaluate a query directly on the representation. While for queries specified in a relational query language it is often possible to find an efficient translation of the query into one on the representation, a database programming language provides richer capabilities, such as executing updates, branching and user interaction that complicate the situation. In the next sections we study the implications of the requirements specified above. We will study criteria for programs to have an observationally deterministic behavior and will provide algorithms for optimizing database programs for set-based execution.

3 Optimizing database programs

The programming model of Section 2 is very powerful and allows for the writing of interesting but also potentially infeasible programs that access the database. We defined the semantics of a database program on uncertain databases to be the one where the program is executed on all worlds in parallel. A direct implementation of this semantics is not possible as uncertain DMBSs often represent compactly a large number of possible worlds. We would therefore like to execute programs in bulk on all worlds at the same time.

Previous work has studied how to translate queries on world-sets into ones on the representation, and in Section 2 we have seen how to do this for updates as well. A database program has richer constructs such as branching and loops and it is not clear how to map those to operations on the representation, as the flow of execution can be different on each world of the world-set. For that we will study techniques for unnesting programs, i.e. mapping programs to a sequence of read and write operations with no branching and loops. Another major issue is that a database program allows users to interact with the database via the read() and write() commands. This can cause problems whenever the user is returned output that is not the same in all worlds or when user is asked to supply different input in each world. Given that uncertain DBMSs often store an exponential number of worlds, such behavior is clearly unacceptable. We will therefore require that the uncertain database be observationally indistinguishable from a complete database (i.e. single-world database). We word this property *observational determinism*: a program is called *observationally deterministic* if the user interaction in terms of input and output of the program in one world is identical to the user interaction in all other worlds of the world-set.

We will first treat the problems of checking for observational determinism and of unnesting updates in isolation. We will start by discussing a method for *statically* checking whether a given program is observationally deterministic using a technique called *c-indicators*. We will then present rules that map update programs to linear sequences of update statements. The techniques from Sections 3.2 and 3.1 will be then used as building blocks of an algorithm that optimizes a database program containing both user interaction and updates.

3.1 Checking observational determinism

A program P is not observationally deterministic whenever it involves different user interaction in each world of the world-set. Let x() be a user interaction operation that appears in P and let Q be the query that binds the values for x(). Then we can reason about whether x() is the same in all worlds by checking whether the query Q produces only certain results. We illustrate the idea with the following examples.

Example 2. Consider for example an uncertain database that stores data as the *or-set relation* of Figure 4. Some fields of the table contain sets, the semantics being that one of the values in the set is the correct one for the respective

R	A	B	C
t_1	{1,2,3}	{4,5}	6
t_2	7	8	9
t_3	10	{11,12,13}	14

Fig. 4. Or-set relation

Let R – relation name, ϕ – boolean condition

Q, Q_1, Q_2 – queries in $RA^+ \cup$ {conf,possible,certain}

$$[\![R]\!] := (R^c, R^u)$$
$$[\![\pi_U(Q)]\!] := (\pi_U([\![Q]\!]^c), \pi_U([\![Q]\!]^u))$$
$$[\![\sigma_\phi(Q)]\!] := (\sigma_\phi([\![Q]\!]^c), \sigma_\phi([\![Q]\!]^u))$$
$$[\![Q_1 \bowtie_\phi Q_2]\!] := ([\![Q_1]\!]^c \bowtie_\phi [\![Q_2]\!]^c, [\![Q_1]\!]^u \bowtie_\phi [\![Q_2]\!]^u)$$
$$[\![Q_1 \cup Q_2]\!] := ([\![Q_1]\!]^c \cup [\![Q_2]\!]^c, [\![Q_1]\!]^u \cup [\![Q_2]\!]^u)$$
$$[\![\text{conf}(Q)]\!] := ([\![Q]\!]^c \cup \text{possible}([\![Q]\!]^u), 0)$$
$$[\![\text{possible}(Q)]\!] := ([\![Q]\!]^c \cup \text{possible}([\![Q]\!]^u), 0)$$
$$[\![\text{certain}(Q)]\!] := ([\![Q]\!]^c, 0)$$

Fig. 5. Propagation of certainty during querying and update operations.

field. In our example the table represents 3 * 2 * 3 = 18 possible worlds over schema $R(A, B, C)$, one for each combination of values in the or-sets. Consider the following three programs that run on R:

```
P1: for($t in "select * from R") write($t);
P2: for($t in "select possible * from R") write($t);
P3: for($t in select certain * from R where A=1
        union select * from R where A <> 1")
      if($t.A=1) write($t);
```

When executing P1 we run into the problem that the output is not the same in all worlds. On the other hand P2 is observationally deterministic, as in each world it outputs the possible tuples, i.e. the tuples that appear in at least one world. For our example the program will print 10 tuples in total: the different versions of t_1, t_2 and t_3. Deciding whether a program is observationally deterministic is not always simple. Consider P3: If we trace the origin of the tuples that are output by this program, we will see that these are exactly the certain tuples with A-value 1 in R. Thus the program satisfies observational determinism. ❙❙

We next present our algorithm for deciding whether a program is observationally deterministic. We first construct a query corresponding to each user interaction (UI) operation in the program, and we couple this with a procedure for deciding whether a query produces only certain results.

Let us suppose that we have annotated the input database and we know which tuples are certain and which are not. More precisely, for each input relation R, we think of R as consisting of two disjoint partitions $R = R^c \cup R^u$, where R^c contains the certain tuples of R, and R^u: the tuples that appear only in

some worlds. R^c and R^u can be computed with the queries $R^c = \text{certain}(R)$ and $R^u = R - \text{certain}(R)$. According to our semantics both queries R^c and R^u produce one result for each world, where the result of R^c is the same in all worlds, and the one of R^u is potentially different.

It is interesting to see how the annotations propagate from the input into the results of querying. For example a selection applied on a relation R can only discard tuples, but cannot make any uncertain tuples certain, and vice versa: no certain tuple will become uncertain. A 'possible' query will make all tuples from the input certain in the result since every world will contain those tuples.

Figure 5 defines an operator $[\![\cdot]\!]$ that takes a query expressed in positive relational algebra extended with the confidence computation predicate and its two special cases: possible and certain, and returns a pair of queries (Q^c, Q^u), whose components compute the certain and the uncertain tuples in the result, respectively. This construction is conservative in the sense that it produces correct results but can omit some on particular inputs. This is the case for queries containing either a projection or a union operation. For example if we apply the projection $\pi_C(R)$ on the world-set represented as the or-set relation of Figure 4, all tuples in the result are certain although the only certain tuple in the input was t_2. Nevertheless, we shall see that for performing static checks, no other construction will produce better results and will be correct on all inputs.

Example 3. The query $Q = \sigma_{A=1}(\text{certain}(\sigma_{A=1}(R)) - \sigma_{A\neq 1}(R))$ corresponds to the write statement of P3 of Example 2. Applying the construction of Figure 5 yields the pair of queries $(\sigma_{A=1}(R^c), \sigma_{A=1 \wedge A\neq 1}(R^u))$. Independent of the actual instance of the database, we can conclude that P3 is observationally deterministic, as the query defining the uncertain partition of the result has an unsatisfiable selection condition and will always return the empty set as result. ❙❙

Using these ideas we can construct a procedure, called *c-indicator*, that "certifies" tuples that are certain in the result of a query.

Ideally we would like to design c-indicators that do not require evaluation of the whole query and then checking which tuples are certain in the output, but instead try to predict this information based on the query and possibly on some constraints that hold on the data. We can measure the quality of a c-indicator based on two criteria. The first one asks for soundness of the c-indicator, that is, that no false positives are produced. The second condition requires that the c-indicator is as close as possible in predicting which tuples are certain in the output. We formalize these requirements below.

Definition 1. We say that a c-indicator $\mathcal{C}$ is *sound* if for all queries Q, databases $\mathbf{A}$ and tuples t, if $\mathcal{C}(Q)(t, \mathbf{A}) = true$, then t is certain in $Q(\mathbf{A})$. A c-indicator $\mathcal{C}$ *dominates* another c-indicator C' ($\mathcal{C} \sqsupset \mathcal{C}'$) iff for all tuples t, queries Q and databases $\mathbf{A}$, if $\mathcal{C}'(Q)(t, \mathbf{A}) = true$, then $\mathcal{C}(Q)(t, \mathbf{A}) = true$, and there is a tuple t such that $\mathcal{C}(Q)(t, \mathbf{A}) = true$ and $\mathcal{C}'(Q)(t, \mathbf{A}) = false$. This means that $\mathcal{C}$ identifies strictly more tuples as being certain than $\mathcal{C}'$. A c-indicator $\mathcal{C}$ is *maximal* iff there is no other c-indicator $\mathcal{C}'$ for the same query Q such that $\mathcal{C}' \sqsupset \mathcal{C}$.

Algorithm 1: Checking observational determinism

```
Input: P: program
Output: true or false
foreach UI operation x() in P do
    Q_x be the query corresponding to x();
    if [[Q_x]]^u is satisfiable then
        return false;
    end
end
return true;
```

There are two obvious solutions to the problem of defining a c-indicator which is sound. The first one is a procedure that rejects all tuples and is therefore trivially guaranteed to produce no false positives. The second way is to enclose the input query Q in a 'certain' construct: certain(Q). This c-indicator will not only be sound but maximal as well. However, deciding tuple Q-certainty is coNP-hard on succinct representation systems [6]. We will therefore study c-indicators that use relational algebra only. We will relax this condition to allow querying certain tuples in the input relations. Intuitively, we can annotate the certain tuples once at the beginning, and then incrementally maintain the annotations when the database is updated.

Theorem 1. *For each query Q expressed in positive relational algebra with possible and certain there exists a c-indicator using relational algebra operators only which is sound and maximal when we assume no knowledge about the database.*

Proof (sketch). Using the construction $[\![\cdot]\!]$ of Figure 5, we define a c-indicator $\mathcal{C}^{pr}$ (pr stands for propagating uncertainty, the idea used in defining $[\![\cdot]\!]$) as the following test: $\mathcal{C}^{pr}(Q)(t, \mathbf{A}) := \{t \Leftarrow [\![Q]\!]^c(\mathbf{A})\}$.

$\mathcal{C}^{pr}$ is sound and maximal. Let $\mathbf{A}$ be an uncertain database, Q be a query over the schema of $\mathbf{A}$ and t be a tuple in the schema of Q. By induction on the structure of the query Q we show that if $t \Leftarrow [\![Q]\!]^c(\mathbf{A})$, then t is certain in $Q(\mathbf{A})$. On the other hand, if $t \notin [\![Q]\!]^c(\mathbf{A})$, then there is a witness world-set $\mathbf{B}$ such that t is not possible in $Q(\mathbf{B})$. If Q is a positive relational algebra query, we can take as witness the world-set $\mathbf{B}_0$ containing one world where all relations are empty.

Finally, using the idea of c-indicators we construct a procedure that automatically decides whether a given program is observationally deterministic or not on all inputs. For each UI operation $x()$ of the given program P we compute the query Q_x that corresponds to $x()$ by composing the for-loop statements that are ancestors of $x()$ in the parse tree of P. Algorithm 1 rewrites Q_x using the c-indicator rules of Figure 5 and checks whether the uncertain partition of that query is satisfiable.

Theorem 2. *Algorithm 1 rejects all programs that are not observationally deterministic.*

3.2 Unnesting updates

We next study how an update program can be turned into a sequence of update statements without for-loops or if-conditions.

Example 4. Consider a modification of the program from Section 1 that satisfies observational determinism:

```
read("Enter license plate, location and color:", $x,$loc,$col);
if (select * from cars where num=$x != NULL) {
  if (select * from cars where num=$x and loc=$loc and color=$col != NULL)
    update cars set wit=wit+1 where num=$x and loc=$loc and color=$col;
  else insert into cars values($x,$loc,$col,1);
}
else insert into cars values ($x,$loc,$col,1);
```

We can linearize the program by mapping the if-else block to the following three update statements:

```
update cars set wit=wit+1
where num=$x and loc=$loc and color=$col;
insert into cars select $x,$loc,$col,1
where not exists (select * from cars where num=$x and loc=$loc and
      color=$col) and not exists (select * from cars where num=$x);
insert into cars select $x,$loc,$col,1
where not exists (select * from cars where num=$x);
```

This program is equivalent to the first one. Using the techniques of Section 2 we can translate it into a program on the representation, which then executes on all worlds at the same time. □

As seen in the above example we can often push if-conditions into the where-clause of an update operation. For this to work however we need to restrict the update such that it does not interfere with subsequent operations necessary to evaluate a query. For example if we exchange the first and the second update statement of the second program we will obtain a different result, although exchanging the if and the else block of the first program does not change its semantics.

We next formalize rules for unnesting update programs. We consider a somewhat simplified version of the API from Section 2, where the control structures are only for-loops and updates are of the following kind. Let R be a relation, $\{A_1, \ldots, A_m\} = \mathbf{sch}(R)$, $c_1, \ldots, c_n$ be constants and ϕ be a condition involving constants and attributes of R. We will restrict ourselves to updates that add tuples of constant values, or change tuples fields to constant values. We consider updates of the form:

update R set $\overline{A} = \overline{c}$ where ϕ;

where $\overline{A} = \overline{c}$ is a shortcut for $A_1 = c_1, \ldots, A_m = c_m$.

Let Q_ϕ denote the semi-join query returning the tuples of R that need to be updated. We will use the notation $U = \{\overline{A} \mapsto \overline{c} \mid Q_\phi\}$ for update operations.

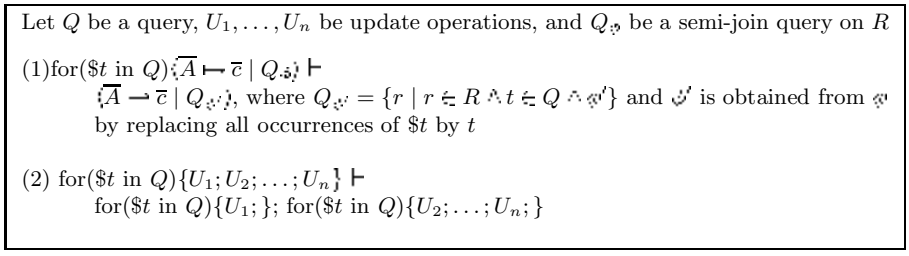


Fig. 6. Rules for unnesting update programs.

To define rules for optimizing programs we rely on the independence of queries from the updates:

Definition 2. Let Q be a query and U be an update operation. For a database $\mathcal{A}$ let $U(\mathcal{A})$ denote the database obtained as a result of executing U on $\mathcal{A}$. We say that Q is *independent* of U iff for any input database $\mathcal{A}$ $Q(\mathcal{A}) = Q(U(\mathcal{A}))$. Similarly, we say that update U_1 is *independent* of another update U_2 if for any input database $\mathcal{A} : U_1(\mathcal{A}) = U_1(U_2(\mathcal{A}))$.

Figure 6 shows two rewrite rules that can be applied iteratively to optimize a database program. Recall that SQL updates have transactional semantics: changes made by update U are not visible to the update condition before the end of the update operation. However, if the update is nested within a for-loop, U will in general be executed multiple times - once for each result tuple returned by Q. Thus changes made by U will be visible in subsequent loop iterations. We next discuss when these rewrite rules produce equivalent programs.

Lemma 1. *(a) If Q_ϕ is independent of the update $U = \{\overline{A} \rightarrow \overline{c} \mid Q_\phi\}$, then rule (1) preserves equivalence.*

(b) If Q_ϕ is independent of U_1, U_1 is independent of $U_i, 2 \leq i \leq n$ and U_i is independent of $U_1, 2 \leq i \leq n$, then rule (2) preserves equivalence.

Proof. Let P and P' denote the programs on the lhs and rhs of rule (1), respectively. Let $r \in R$ be a tuple that is updated in P and let this for tuple t in Q. Then $r \in Q_{\phi(t)}(\mathcal{A}')$ where $\phi(t)$ is the condition obtained by substituting the variable t in ϕ with the values from t and $\mathcal{A}'$ is the state of the database at the beginning of that iteration of the loop. Since Q_ϕ is independent of U, $Q_\phi(\mathcal{A}') = Q_\phi(U(\mathcal{A}')) = Q_\phi(\mathcal{A})$, where $\mathcal{A}$ is the initial database before the start of P. But then this is equivalent to $r \in Q_\phi(\mathcal{A})$. Thus r is also updated in P', and P and P' are equivalent.

(b) Let P and P' be the programs on the lhs and rhs of the rule, respectively. Suppose U_i is updating relation R_i with $R_1, \ldots, R_n$ not necessarily disjoint. Since Q is independent of U_1, each loop in P' performs exactly the same iterations as the loop in P. In addition, since U_1 is independent of $U_2, \ldots, U_n$, a tuple $r \in R_1$ is updated in P iff it is updated in P'. Similarly, since each U_j, $2 \leq j \leq n$ is independent of U_1, a tuple $r \in R_j$ is updated in P iff it is updated in P'.

Algorithm 2: Optimize programs

Input: P: program
Output: P': program equivalent to P that can be executed on all worlds or FAIL.
foreach *maximal subtree* P_0 *of P with no UI operations* **do**
  Let L:=unnest(P_0);
  return *'FAIL' if* P_0 *cannot be unnested*;
  Replace P_0 by L in P;
end
return *'FAIL'* if P is not observationally deterministic;
Otherwise **return** P;

The correctness proof for Rule (2) does not use the fact that attribute values can be changed to constants only. Under the independence assumptions of Lemma 1 the rule remains correct when updates can change fields to arbitrary values, not only constants. We next discuss the implications for Rule (1) of allowing variables to appear in the set list. Consider for example the following update block, where x_i is either a constant or a reference of the form $\$t.A$:

$$\text{for}(\$t \text{ in } Q)\{\text{update R set } A_1 = x_1, \ldots, A_m = x_m \text{ where } \phi\}$$

In such an update block it is possible that the same field is set to two different values in two different loop iterations. Hence the final value of the field will depend on the order of reading the for-loop tuples, which is often undesirable. We require that this never happens, that is, a field is always set to the same value or is left unchanged. We can then generalize rule (1) for the case where variables can appear in the set-clause of an update statement.

Checking independence of queries from updates. The problem of statically deciding whether a query is independent from an insertion or deletion update has been studied in [5] and [7]. The proposed solution consists in checking independence of two programs: one that computes the query answer before the update, and one - after the update. In our setting we are also considering updates specified with an SQL update statement. Since those can be simulated with a pair of insert/delete, one can reduce the problem of deciding independence from a general update statement to independence of insertion and deletions. Note however that we often need to check independence of the query corresponding to the where clause of an update statement from the update itself. Thus deciding independence at the level of inserts and deletes will be unnecessarily restrictive, as it will always return a negative answer. We can use a simpler but more precise condition to check update independence, namely: if an attribute appears both on the left side of the set-clause and in the where-clause of an update statement, then the update query is in general not independent of the update.

3.3 Rewriting database programs for bulk execution

We will consider database programs where both updates and user interaction commands can be nested. Let P be the parse tree for a program, where P's

nodes are sequences of for-loops, update statements and user interaction (UI) commands. Algorithm 2 shows an algorithm that optimizes a program for bulk execution on all worlds, or returns 'FAIL' if no optimization is found. On success the algorithm returns a program that satisfies the following two conditions:

1. All update operations are on the top-level or are nested within loops that operate on certain query results only.
2. The program is observationally deterministic.

The algorithm considers all maximal rooted subtrees of the parse tree for P that do not contain any UI operations. For those the algorithm applies the unnesting techniques (presented in the previous subsection) to turn them into flat sequences of update statements. Let P' be the program that results from this step. If we can verify that P' is observationally deterministic, then P' is the result of the optimization procedure. Finally, we can state our main result:

Theorem 3. *If Algorithm 2 returns program P', P' is equivalent to the input program P, satisfies observational determinism, and all update operations are either on the top level or are nested in for-loops that iterate over certain results.*

References

1. L. Antova, T. Jansen, C. Koch, and D. Olteanu. "Fast and Simple Relational Processing of Uncertain Data". In *Proc. ICDE*, 2008.
2. L. Antova, C. Koch, and D. Olteanu. "From Complete to Incomplete Information and Back". In *Proc. SIGMOD*, 2007.
3. O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. "ULDBs: Databases with Uncertainty and Lineage". In *Proc. VLDB*, 2006.
4. N. Dalvi and D. Suciu. "Efficient query evaluation on probabilistic databases". In *Proc. VLDB*, 2004.
5. C. Elkan. Independence of logic database queries and updates. In *Proc. PODS*, 1990.
6. T. Imielinski and W. Lipski. "Incomplete information in relational databases". *Journal of ACM*, **31**(4), 1984.
7. A. Y. Levy and Y. Sagiv. Queries independent of updates. In *Proc. VLDB*, 1993.
8. C. Re, N. Dalvi, and D. Suciu. "Efficient Top-k Query Evaluation on Probabilistic Data". In *Proc. ICDE*, 2007.
9. P. Sen and A. Deshpande. "Representing and Querying Correlated Tuples in Probabilistic Databases". In *Proc. ICDE*, 2007.
10. M. Soliman, I. Ilyas, and K. C. Chang. "Top-k Query Processing in Uncertain Databases". In *Proc. ICDE*, 2007.

Towards Special-Purpose Indexes and Statistics for Uncertain Data*

Anish Das Sarma, Parag Agrawal, Shubha U. Nabar, Jennifer Widom

{anish,paraga,sunabar,widom}@cs.stanford.edu
Stanford University

Abstract. The Trio project at Stanford [35] for managing data, uncertainty, and lineage is developed on top of a conventional DBMS. Uncertain data with lineage is encoded in relational tables, and Trio queries are translated to SQL queries on the encoding. Such a layered approach reaps significant benefits in terms of architectural simplicity, and the ability to use an off-the-shelf query processing engine. In this paper, we present special-purpose indexes and statistics that complement the layered approach to further enhance its performance.
First, we identify a well-defined structure of Trio queries, relations, and their encoding that can be exploited by the underlying query optimizer to improve the performance using Trio's layered approach. We propose several mechanisms for indexing Trio's uncertain relations and study when these indexes are useful. We then present an interesting order, and an associated operator, which are especially useful to consider when composing query plans. The decision of which query plan to use for a Trio query is dictated by various statistical properties of the input data. We identify the statistical data that can guide the underlying optimizer, and design histograms that enable estimating the statistics accurately.

1 Introduction

The field of uncertain databases has received considerable attention for several decades now. While most prior work (e.g., [1, 5, 12, 16, 18, 24–27]) focuses on theoretical aspects, there has been recent interest in building systems [4, 8, 10, 29]. Motivated by a diverse set of applications such as data integration, deduplication, scientific data management, information extraction, and others, the Trio project at Stanford [29, 35] has been studying the combination of uncertainty and lineage as the basis for a new type of database management system.

In this paper, we study techniques for enhancing query optimization in Trio. The basic construct for uncertainty in Trio's ULDB data model [6] is *alternatives*. Alternatives in a tuple specify a nonempty finite set of possible values for the tuple. For example:

(Thomas, Main St.) || (Tom, Maine St.)

* This work was supported by the National Science Foundation under grants IIS-0324431 and IIS-0414762, by grants from the Boeing and Hewlett-Packard Corporations, by a Microsoft Graduate Fellowship, and by Stanford Graduate Fellowships from Cisco Systems and Sequoia Capital.

contains a tuple with two alternatives giving the two possible values for the tuple. The ULDB data model is defined formally in Section 2. The Trio system is layered on top of a conventional relational DBMS [29]. ULDB relations are encoded as conventional relational tables, and a rewriting-based approach is used for most data management and query processing.

The simple and elegant layered approach in Trio enables using an off-the-shelf query processing engine[1] (QPE) and its optimization capabilities. However, the performance of the layered approach can further improve if the QPE detects and exploits the inherent "structure" of encoded ULDB relations during query processing. This paper suggests novel techniques that augment the optimization capability of the layered approach's QPE.

We show that the structure of ULDB relations merits building specialized index structures and associated access methods. While some of these index structures are equivalent to conventional indexes over encoded ULDB relations, some others are non-traditional indexes that cannot be created by a traditional QPE. We then show that query plans executing queries over Trio relations need to consider a special *interesting order*, namely that of grouping alternatives based on the tuple they are a part of. Consequently, we need an operator that performs the grouping. Such a grouping through the query plan eliminates the need for an expensive sort operation currently performed on the result in Trio.

Every Trio query can be answered using many query plans assembled from the new operators, indexes, and access methods mentioned above in conjunction with those already existing in the QPE. As in conventional query optimization, the choice of which query plan to use is based on an estimate of the cost of executing query plans, which critically depend on various forms of statistics and cardinalities maintained by the database. We enumerate several important statistics, and provide histograms that allow us to efficiently and accurately estimate the statistics. The techniques describe in this paper can be incorporated into the underlying QPE, while still maintaining the overall layered architecture adopted by Trio.

Finally, we present several interesting avenues for future work that our paper suggests. Although several systems have been built recently for managing uncertain data, none of them incorporate query optimization techniques similar to the ones described in this paper. Prior work in Trio itself has also focussed on other aspects such as modeling and design, confidence computation, and versioning. Obviously, query optimization in conventional databases is an extensively studied area, and we believe this first paper on query optimization for uncertain databases can form the basis for a lot of further work. Since many previously proposed data models for uncertainty (for example, [3, 5, 12, 17, 28, 32]) use similar constructs as ULDBs, the techniques described in this paper can be suitably adapted for these data models as well.

1.1 Contributions and Outline

The specific contributions of this paper are as follows:

[1] Our current implementation uses PostgreSQL

1. We provide techniques for indexing ULDB relations encoded as conventional relations, and describe the new access methods they yield. (Section 3)
2. We motivate the need for considering a new interesting order in query plans, and design an operator that ensures this order. (Section 4)
3. We enumerate various kinds of statistics necessary in choosing the optimal query plan from the set of all query plans combining conventional and new operators described above. We present histograms that enable estimating these statistics efficiently and accurately. (Section 5)
4. We discuss a slew of interesting and challenging problems our paper opens up, that we hope would form the basis for further research in the area. (Section 6)

Section 2 introduces our data model and its relational encoding, Section 7 presents related work, and we conclude with future work in Section 8.

2 Preliminaries

We first introduce Trio's ULDB data model, then present the relational encoding of ULDBs in the layered approach, and finally describe the workload of queries we consider.

2.1 ULDB Data Model

We briefly introduce the ULDB data model here, and refer the reader to [6] for additional features and detailed semantics. Each tuple in a ULDB relation consists of a set of mutually-exclusive *alternatives*, each of which can be the tuple's actual value. ULDB relations conform to the standard *possible-worlds semantics* [1, 6, 11, 31, 33]: A ULDB relation represents a set of possible worlds, each of which is an ordinary relation. The possible worlds for an uncertain relation are obtained by choosing one alternative value for each tuple, in all possible ways. For example, the following ULDB relation represents four possible worlds. (Alternatives are separated by ||.)

R(Name, Address)
(Thomas,Main St.) || (Thomas,Maine St.)
(Bill,Poplar Ave) || (William,Poplar Ave)

The ULDB data model also has other uncertainty constructs, such as "?" and confidence values, which are not crucial for optimization. It's important to note that the Trio system decouples confidence computation from data computation [14], and hence confidence values are disregarded for the rest of the paper. Also, we don't discuss the lineage feature in ULDBs (refer to [6]), as lineage is only a function of the query, and independent of the specific query plan used to compute the result.

2.2 Relational encoding

We now describe how the restricted ULDB data model described above is encoded in regular relational tables. We refer the reader to [7, 29] for complete details on encoding

ULDB databases. Hereafter we use *x-tuple* to refer to a tuple in the ULDB model, which includes alternatives, and we use *tuple* to denote a regular relational tuple.

Consider a ULDB relation $T(A_1, ..., A_n)$. The data portion of T is stored as a conventional table referred to as T_{enc} with two additional attributes: $T_{enc}(\text{aid}, \text{xid}, A_1, \ldots, A_n)$. Each alternative in the original ULDB table is stored as its own tuple in T_{enc}, and the additional attributes function as follows:

- **aid** is a unique alternative identifier.
- **xid** identifies the x-tuple that this alternative belongs to.

Relation R from the previous section would be encoded as follows.

R_{enc}

xid	aid	Name	Address
1	11	Thomas	Main St.
1	12	Thomas	Maine St.
2	13	Bill	Poplar Ave
2	14	William	Poplar Ave

2.3 Queries

As mentioned before, queries over ULDB relations are translated to queries over the encoded relations. To obtain `xid`'s on the resulting relation, the alternatives of the result need to be grouped based on which x-tuple they are a part of. Therefore, all tuples in the result of the translated query are grouped by xids of the input relations. For example, if we perform a join of R and S, the translated query over R_{enc} and S_{enc} includes the clause "`group-by` R_{enc}`.xid,` S_{enc}`.xid`". Exact details of this translation are omitted, and can be found in [7, 29].

3 Indexing ULDB Relations

As described in Section 2, there is a special attribute `xid` in all Trio encoded relations, and all query translations involve a group by on `xid`. In this section, we introduce new indexes that are especially useful in the presence of this special attribute. We use a simple running example to ground our discussion.

Consider a relation `R`, and a selection query which does a range scan on the attribute `A`: "`select * from R where A` $\leq$ `5`". The translated query groups the result by the `xid` attribute. The relation R_{enc} is often stored clustered on `xid`, but may also be clustered on other attributes, which may or may not be `A`. For the selection query on `A`, the following indexes may be useful:

- Index on `A`
  An index on `A` may be used to retrieve only the tuples in R_{enc} that satisfy the predicate `A` $\leq$ `5`. This index lets us efficiently retrieve all alternatives that satisfy the predicate, but now they need to be grouped to form x-tuples. A sort on `xid` is required to group the result alternatives into x-tuples. Such a query plan can be efficient for highly selective queries, i.e., the result contains very few alternatives, making the grouping step inexpensive.

– Index on `xid`
  An index on `xid` lets us retrieve all alternatives in an x-tuple together; i.e., it returns tuples of R_{enc} in order of their `xid` values. This allows us to avoid the sort since result tuples are generated already grouped by `xid`. This index may be useful if the predicate is not very selective, especially if the data is stored clustered by `xid`.
– Index on `(xid,A)`
  If x-tuples are very wide, i.e., contain a large number of alternatives, we may be able to use an index on (`xid, A`), to only retrieve alternatives that satisfy the predicate. This also avoids the sort at the end, and thus may yield an efficient execution plan.
– Index on `(A,xid)`
  For queries that use an equality predicate on `A`, it might be useful to use an index that returns all alternatives satisfying the predicate grouped by `xid`. This index allows us to avoid the sort which may be expensive for large results. But equality predicates seldom have large results. We would ideally like an index that also works for range queries, and still avoids the sort. Such an index is presented next.

The indexes discussed above help in either avoiding the sort required for the group-by on `xid`, or prune down the amount of data accessed by evaluating the predicate before retrieving the tuples. An index on `(A,xid)` accomplishes both objectives for equality predicates on `A`. We now describe a new index that generalizes this index for efficient range scans over relations stored clustered by `xid`.

– Index on A_x
  An index on A_x refers to an index that retrieves all x-tuples that contain an alternative satisfying some predicate on `A`. We can then apply the predicate to each alternative of the x-tuple to keep only those that satisfy it. Although this index may often retrieve more tuple alternatives than an index on `A`, it can often be more efficient because it makes no more random accesses, and also avoids the need for a sort. The sort is avoided because the index guarantees an "interesting order" that has result alternatives grouped by `xid`.
– Index on `(`A_x`,A)`
  For wide x-tuples, it may again be useful to have an index on `A` within an x-tuple. This index is similar to the index on `(xid,A)`, but prunes x-tuples and retrieves only those that contain at least one alternative that satisfies the predicate.

Example 1. We illustrate the benefit of A_x over the conventional indexes described before using an example. Again consider the query: "`select * from R where A` $\leq$ `5`". Suppose R contains a alternatives satisfying the predicate, and suppose these a alternatives constitute x x-tuples. Let us consider the cost of executing this query using four of the relevant indexes:

1. Index on `A`: We would use the index to get all the a alternatives using index lookups, and then these a alternatives would need to be grouped in memory based on their `xid`s. Hence the total cost is: a index lookups + group a tuples.
2. Index on `(A,xid)`: Since the query involves a range scan, using the additional index on `xid` does not help us do the grouping along with the index lookups. Hence, the cost by using the index on `(A,xid)` is the same as the cost by using the index on `A`.

3. Index on `xid`: Using the index `xid` we scan the entire relation, and within each x-tuple we check whether there is any alternative with $A \leq 5$. The cost of this would be to do an index scan of the entire relation. The grouping step is saved because all alternatives are obtained grouped by `xid`s.
4. Index on A_x: Finally consider using the index on A_x. We perform an index scan to obtain all x-tuples containing at least one alternative with $A \leq 5$. The cost of this step is an index scan of x x-tuples. We then need to filter all alternatives among these x-tuples that do not satisfy the predicate. Hence using the index A_x, we save the grouping step and still require looking only at x-tuples that have alternatives satisfying the predicate.

□

4 Query Plans

In this section, we discuss some new challenges that are encountered in creating query plans to answer queries over uncertain data. We start by discussing a new "interesting order" that can benefit many queries.

4.1 XGroup

Recall the index on A_x described above, which allows for an efficient access method to retrieve alternatives grouped by `xid`. For complicated queries, possibly including several joins, it may be useful to maintain this grouped order incrementally all through the query plan in order to avoid a massive group-by at the end. Intuitively, the idea is to have x-tuples flowing through operators instead of alternatives. By maintaining all alternatives of an x-tuple (nearly) together, we may get efficient executions of queries with large results. We now formally define the XGroup order. We shall see that XGroup is a fundamentally weaker requirement than sort on an attribute, and hence can often be maintained over the entire query plan without significant overhead.

Definition 1 (XGroup). *Let $\mathcal{R} = \{R_1, \cdots, R_n\}$ be the relations in the* `from` *clause of a query. The translated query contains a* `group-by` *on the* `xid` *attributes of each relations in $\mathcal{R}$. Suppose the subtree rooted at a node N in the query tree joins relations in $\mathcal{R}_N \subseteq \mathcal{R}$. If the tuples flowing out of N are guaranteed to be grouped by the* `xid`*s of a set $\mathcal{B}_N \subseteq \mathcal{R}_N$ of the relations, then N is said to guarantee the XGroup order on the set of relations $\mathcal{B}_N$. The final query result thus needs to be XGrouped on $\mathcal{R}$.* □

An index access on relation R using the index on A_x for some attribute `A` guarantees an XGroup on {`R`}. Other access methods that use an index on `xid` or `(xid,A)`, or that scan relations clustered by `xid` also guarantee an XGroup on {`R`}. Note that selection and projection operators also preserve XGroups; i.e., if the input to such an operator is XGrouped on $\mathcal{B}$, the output will also be XGrouped on $\mathcal{B}$.

Many join operators also preserve XGroups. For instance, in a nested join operator, suppose the inner set of input alternatives are XGrouped on $\mathcal{B}_i$ and the outer are XGrouped on $\mathcal{B}_o$. It is easy to see that operators like nested-loop join (or nested-block

joins) and nested-loop index join preserve the XGroup on both the inner and the outer, i.e., the result alternatives will be XGrouped on $\mathcal{B}_i \cup \mathcal{B}_o$. Some other join operators may only preserve XGroups for one of the inputs. For instance a hash-join preserves the XGroup of the outer, i.e., returns alternatives XGrouped on $\mathcal{B}_o$. Many operators in traditional database query processors naturally preserve XGroup, making it efficient to maintain and utilize.

Recognizing and incrementally maintaining XGroups through a query plan can often help avoid an expensive `group-by` (usually implemented through a sort) at the end. As an extreme example, if all relations in a query are accessed using access methods that guarantee XGroups on input relations, and all operators preserve them, then no final `group-by` needs to be performed in the query plan. Our selection example earlier was the simplest case that illustrates how the `group-by` can be eliminated. Requiring all access methods and operators to preserve XGroups narrows the set of acceptable query plans to a very small set. Hence, we allow query plans to have combinations of access methods and operators that may or may not preserve XGroups. However, we can still benefit from partially XGrouped results in making the final `group-by` cheaper, through a new unary operator described next.

Definition 2 (XGB). *Let the input to operator XGB be alternatives XGrouped on $\mathcal{B}_i$. XGB returns the alternatives XGrouped on $\mathcal{B}_o \supseteq \mathcal{B}_i$.* ❙❙

The XGB operator is thus XGroup-aware ensuring that the result alternatives are XGrouped on a larger set of relations. It essentially breaks up each group in the input corresponding to one combination of `xid`s of $\mathcal{B}_i$ into even smaller groups, corresponding to combinations of `xid`s of $\mathcal{B}_o$. The advantages of using this new operator are threefold: (1) In conjunction with an access method that provides the XGroup order, the XGB operator allows us to incrementally do the `group-by`. This may have lower cost than doing a single expensive `group-by` at the end of the query plan. (2) The operator only needs to look at one group in the input stream of alternatives at a time to construct the output grouping. If the groups in the input are not large, XGB can be performed very efficiently, operating in a non-blocking fashion. (3) For the same reason, the memory requirements for performing XGB is also considerably less than an operator that XGroups on $\mathcal{B}_o$ assuming no input grouping.

Example 2. Consider a query that joins two relations R_1 and R_2 with n_1 and n_2 x-tuples respectively. Consider a query plan that joins alternatives from the two relations in any arbitrary order and then does a final `group-by` on R_1.`xid`, R_2.`xid`. The cost of this final `group-by` will be $C_G(n_1 n_2)$, where C_G gives the expected cost of doing the `group-by` as a function of the total number of x-tuples. For instance, if the `group-by` is achieved by sorting the alternatives by R_1.`xid`, R_2.`xid` then $C_G(n_1 n_2)$ would be $\mathrm{O}(n_1 n_2 \log(n_1 n_2))$. In contrast, if the output of the join is already XGrouped on R_1.`xid`, the cost of the final `group-by` is $\mathrm{O}(n_1 C_G(n_2))$, which is typically less than $C_G(n_1 n_2)$. Intuitively, a problem of size $n_1 n_2$ is reduced n_1 problems of size n_2. ❙❙

The presence of even one access method or operator that guarantees the XGroup order can thus reduce the cost of the `group-by`, if the optimizer recognizes the operators preserving XGroups.

4.2 Query Planning

In order to choose an optimal plan for executing a query, the query planner needs to decide which combination of the new indexes, operators introduced, and the traditional relational operators must be used for a given query. We motivate the need for estimating various statistics that would guide the query planner in choosing an efficient query plan.

Consider our example query from Section 3: "`select * from R where A` $\leq$ `5`". We now look at each access method discussed in Section 3 emphasizing the relevant statistics that impact their execution cost.

- Index on `A`
  To estimate the cost of accessing the relation through an index on `A`, the optimizer needs to determine the number of alternatives that satisfy the predicate.
- Index on `xid`
  To estimate the number of x-tuples retrieved using an index on `xid`, the optimizer needs to know the total number of x-tuples in the relation `R`.
- Index on `(xid,A)`
  As discussed in Section 3, this index is useful only if x-tuples contain large number of alternatives. Hence, to decide whether or not to use this index, an estimate indicating the average number of alternatives per x-tuple (width of x-tuple) is useful. The average width of x-tuples satisfying the predicate might differ from the average width for the entire relation, and can yield more accurate cost estimates.
- Index on `(A,xid)`
  The number of alternatives that satisfy the predicate determines the cost of accessing the relation using an index on `(A,xid)`. In addition, the number of x-tuples returned is determined by estimating the number of x-tuples that contain at least one alternative that satisfies the predicate on `A`.
- Index on A_x
  The number of random accesses made in accessing the relation `R` using an index on A_x is determined by the number of x-tuples that contain at least one alternative satisfying the predicate.
- Index on (A_x`,A`)
  This index is useful in cases where the average width of an x-tuple is large, thus justifying indexing on `A` within an x-tuple. A useful statistic to determine the efficacy of the index is the average width of x-tuples that satisfy the predicate.

In the next section, we formally define the above statistics and provide methods for estimating them.

5 Statistics and Histograms

We will now formally define the different statistics from the previous section that guide the optimizer in choosing the optimal query plan. Some of these statistics can be exactly maintained, while others need to be estimated and we introduce new kinds of histograms for this purpose.

5.1 Exact Cardinalities

For each relation R in the database, the following global statistics can be exactly maintained:

- X-Card(R): The number of x-tuples in R. In terms of R's encoding, X-Card(R) gives the number of distinct `xid`s in R_{enc}.
- A-Card(R): The number of alternatives in R. A-Card(R) translates to the number of tuples in R_{enc}.
- AvgWidth(R): The average number of alternatives per x-tuple in R, i.e., $\frac{\text{A-Card}(R)}{\text{X-Card}(R)}$.

Example 3. Consider the following relation R.

R(Name, Address, Salary)
(Thomas, Main St., 50K) || (Thomas, Maine St., 50K)
(Bill, Poplar Ave, 35K) || (William, Poplar Ave, 40k) || (Billy, Poplar Ave, 40K)
(Alice, Euclid Ave, 10K)

Here X-Card(R) = 3, A-Card(R) = 6 and AvgWidth(R) = 2. ❙❙

Next we consider statistics that are infeasible to maintain exactly, and hence need to be estimated.

5.2 Alternative Counts

Consider a relation R with an attribute A. The following are statistics about alternatives that we would like to maintain for point and range queries:

- A-Selectivity(R,A,v): The number of alternatives in R that satisfy $A = v$.
- A-Selectivity(R,A,x,y): The number of alternatives in R that satisfy $x \leq A \leq y$.

Example 4. Consider the relation R from Example 3. A-Selectivity(R, Name, Thomas) = 2, whereas A-Selectivity(R, Salary, 40K, 60K) = 4. ❙❙

Clearly, the above selectivities translate to counting the number of tuples in R_{enc} satisfying $A = v$ and $x \leq A \leq y$ respectively. These cardinalities over the conventional relational table R_{enc} can be estimated using well-known sampling or histogram techniques. For example, in Trio we can build a histogram over R_{enc}: The histogram consists of buckets corresponding to the range of all possible values in A. A histogram bucket with bucket boundary $[p, q]$ maintains the count of the number of tuples in R_{enc} that satisfy $p \leq A < q$. Now, we can use the bucket frequencies to estimate A-selectivities by making standard uniformity assumptions.

Example 5. For relation S_{enc} with integer attribute A, suppose the histogram on A has bucket intervals 0, 5, 10, and so on; i.e., bucket $B_{0,5}$ stores the number of tuples in S_{enc} with $0 \leq A < 5$, $B_{5,10}$ for $5 \leq A < 10$, etc. The number of tuples in S_{enc} satisfying $3 \leq A \leq 8$ is estimated as $\frac{2}{5} B_{0,5} + \frac{4}{5} B_{5,10}$. ❙❙

5.3 X-tuple Counts

For relation R with attribute A, we would also like to estimate the number of x-tuples in R that contain some alternative satisfying a point or range predicate on A:

- X-Selectivity(R,A,v): The number of x-tuples in R containing at least one alternative satisfying $A = v$.
- X-Selectivity(R,A,x,y): The number of x-tuples in R containing at least one alternative that satisfies $x \leq A \leq y$.

Example 6. Consider relation R from Example 3 again. Here X-Selectivity(R, Name, Thomas) = 1 and X-Selectivity(R, Salary, 40K, 60K) = 2. □

Estimating X-Selectivity(R,A,v): Just as for estimating A-selectivities, here too we can build a histogram that now counts the number of x-tuples instead of the number of alternatives. No existing histograms can be used, however, and we build new kinds of histograms for this purpose.

Let us first attempt constructing a histogram by creating buckets on attribute A, and we shall then improve on this histogram. Within a bucket $[p, q]$, we store the number of x-tuples that contain at least one alternative satisfying $p \leq A < q$. Note that, unlike in the case of the histogram for A-selectivity whose buckets counted distinct alternatives, in this histogram a single x-tuple may contribute to the count in multiple buckets. For instance, if an x-tuple contains an alternative with A in range $[p_1, q_1]$ as well as another alternative with A in range $[p_2, q_2]$, then the x-tuple will contribute to the counts in both the buckets. Given a value v, suppose v corresponds to bucket $[p, q]$, i.e., $p \leq v < q$, then we estimate X-selectivity(R,A,v) to be $\frac{B_{p,q}}{(q-p)}$, where $B_{p,q}$ is the stored count in bucket $[p, q]$.

Note that the above estimate for a specific value v assumes that if an x-tuple is in bucket $[p, q]$, the probability of it having an alternative with value v, where $p \leq v < q$ is $\frac{1}{(q-p)}$. However, we can refine this probability if we know the number of alternatives in an x-tuple. Suppose an x-tuple has k alternatives, whose A values are randomly and independently distributed between p and q, the probability of having at least one alternative with value v is $1 - (\frac{q-p-1}{q\ \ p})^k$. (If we know that the distinct A values in the alternative are independently and randomly distributed in $[p, q)$, then we replace k with the number of distinct A values in the x-tuple.)

We can use the observation above to enhance our histogram. We construct a 2-D histogram with A as one dimension, and the size $\mathcal{S}$ of the x-tuple (either as the number of alternatives or the number of distinct A values, based on the distribution described above) as the second dimension. In a bucket $[p, q]$ for dimension A, and $[n_1, n_2]$ for dimension $\mathcal{S}$, we store the number of x-tuples having either: (1) at least k alternatives with A value in the range $[p, q)$, or (2) alternatives with at least k distinct A values in the range $[p, q)$. Recall, we use (1) above if we assume the distribution of all alternatives is independent, and we use (2) if we assume the distribution of distinct A values is independent.

Estimating X-Selectivity(R,A,x,y): While at first glance it might seem that we can use the histogram described above to estimate X-selectivity(R,A,x,y) also, that is not the case. We would get an incorrect estimate using the histogram described above because each x-tuple contributes to the count in multiple buckets. [2] Hence if we aggregate counts over multiple buckets, we would end up double-counting x-tuples corresponding to multiple buckets, as shown by the following example.

Example 7. Suppose for relation S, the histogram on attribute A is bucketed at multiples of 5. If we want an estimate of X-Selectivity(S,A,0,9), adding up the counts corresponding to buckets $[0, 5]$ and $[5, 10]$ gives an incorrect answer, because x-tuples could contain both alternatives with A values in $[0, 5)$ as well as A values in $[5, 10)$. Hence the returned sum would be an overestimate of the actual count. □

We need to create a more complex histogram to be able to give accurate expected estimates. We use a 3-D histogram with dimensions A_{min}, A_{max}, and size $\mathcal{S}$. A_{min} corresponds to the minimum A value for an alternative in an x-tuple, A_{max} corresponds to the maximum A value. As before $\mathcal{S}$ corresponds to the size of the alternatives falling in the bucket, where size is given either by the number of alternatives or by the number of distinct A values, depending on the distribution assumption. In the histogram bucket corresponding to range $[p, q]$ for A_{min}, $[r, s]$ for A_{max}, and $[n_1, n_2]$ for $\mathcal{S}$, we store the number of x-tuples with size $\mathcal{S}$ in range $[n_1, n_2)$ that have the minimum A value in range $[p, q)$, maximum A value in range $[r, s)$. Note that now every single x-tuple corresponds to exactly one bucket of the histogram. Hence, we can estimate X-Selectivity(R,A,x,y) by adding up the contributions from all relevant buckets without any danger of double-counting.

5.4 Other Statistics

Finally, we note that several other kinds of statistics can be estimated using the estimation techniques described above. For example, suppose for relation R we want to the know the average number of alternatives satisfying $A = 5$ among x-tuples that contain at least one alternative with $A = 5$, we can compute the average as $\frac{A-selectivity(R,A,5)}{X-selectivity(R,A,5)}$.

Another similar statistic worth mentioning that can be estimated using techniques described earlier is finding the total or average number of alternatives in R in x-tuples that contain at least one alternative satisfying a predicate:

– A-Average(R,A): The average number of alternatives in x-tuples that contain at least one alternative satisfying $A = v$ or $A \in [x, y]$.

For A-Average we can use a histogram similar to the one for X-Selectivity, but by eliminating the size dimension. We can then store the total count of alternatives from the x-tuples in each bucket, instead of storing the number of x-tuples.

[2] By 'incorrect', we mean that the estimator will not be an unbiased estimator for X-Selectivities, even under standard uniformity assumptions.

6 Discussion and Future Work

In this section we discuss important issues arising from this paper that we do not address, as well as more general directions for the future that our work suggests.

Construction and Maintenance: The first question that arises about our techniques presented in Section 5 is how we construct the histograms that enable estimating the various statistics. Construction of a histogram entails deciding the bucket boundaries for all the dimensions followed by populating the counts in each bucket. Recall that our histograms for ULDB relations translate to statistics on the encoded conventional relations. Hence we can leverage previously proposed techniques for determining the bucket boundaries and constructing the histogram through sampling [9, 30]. Alternately, we could scan the entire relations apriori and compute exact frequencies for all buckets.

The second issue that arises is that of incrementally maintaining histograms as data is modified. While at first glance it seems like we can still use traditional techniques for maintaining histograms [23], we plan to explore whether there are more efficient techniques that can be applied specific to the kinds of data modifications Trio supports. [13]

Operators and Indexes: An interesting followup of our work would be to study whether there are any other specialized operators and indexes that would be useful for query execution in Trio. One possibility we considered is to combine the grouping operator we have proposed with other operators such as various forms of join operators. However, this does not seem to provide any obvious additional benefit over using the join and grouping operator in sequence. (For instance, combining XGB with a hash join might on one hand perform fewer operations, but on the other hand require more memory.) Detailed study of combining groups of operators to form a single more efficient operator is left as future work.

Another issue that we have not addressed is the problem of automatically deciding the indexes to be created given a specific instance of a database and workload of queries, in the spirit of design advisers provided by most commercial DBMSs.

System: We intend to implement the new operators and techniques presented in this paper, and incorporate them into the Trio system. We then plan to perform an extensive experimental evaluation quantifying the benefits these techniques add to the current layered approach.

Statistics Propagation: We have proposed techniques for estimating statistics for input relations. These statistics can be used in conjunction with conventional sampling-based approaches [2, 15] to estimate certain statistics, alternative and x-tuple cardinalities, for intermediate results output by operators in a query plan. However, reusing other recent techniques such as sketches [20] and wavelets [21, 22] for estimating all statistics of intermediate results to uncertain databases forms an interesting direction of future work.

Other Models and Systems: In this paper we've presented techniques that are motivated by the Trio system and its data model. However, some of our techniques can be

adapted for other data models that use similar constructs [3, 5, 12, 17, 28, 32]. More generally, it would be interesting to explore operators and associated estimation techniques for other uncertain database systems. The grand goal of building a generic optimizer for uncertain databases poses several challenging problems including efficient indexing, estimating statistics, constructing and costing query plans, and much more.

7 Related Work

The study of uncertainty in databases goes back to the early 80s. A large body of previous work has been theoretical in nature and a very small subset of it can be found in [1, 5, 12, 16, 18, 24–27, 36]. Recently there have been several efforts in building systems for managing uncertainty [4, 8, 10, 29]. While most of these projects have studied efficient algorithms for query processing, none of them actually addresses the problem of query optimization through specialized indexes, operators, and statistics estimation, which are the subject of our paper.

Two notable pieces of recent work studying query processing in probabilistic databases are [11, 14], whose focus is on probability computation. Reference [11] characterizes when a query can be computed using a *safe plan*, which can be very efficiently executed. Reference [14] studies how lineage in Trio can aid in making confidence computation more efficient. Importantly, it shows that lineage allows us to decouple data and confidence computation in Trio, enabling us to use any query plan for data computation. It is this decoupling that allows the optimizer to consider any query plan, without worrying about confidence computation.

Reference [34] proposes interesting new techniques for indexing uncertain data with probabilities. The indexes proposed in [34] are different from ours and focus on the probabilities. Their indexes are very useful for queries with thresholds on probabilities. However, as described above, Trio adopts lineage-based probability computation, and hence we focus only on data computation in this paper. We have proposed new indexing mechanisms specific to the ULDB data model and its relational encoding, which are more useful for Trio query processing. Finally, [34] does not consider the problem of estimating various kinds of statistics on uncertain data.

Obviously there has been a huge body of work studying every aspect of query optimization for conventional databases. Since there have been many papers written on every topic, including relational operations, indexing, histograms, statistics, and query plan selection, we do not review this literature here. We refer the interested reader to any standard database textbook, such as [19].

8 Conclusions

We argued that Trio's current architecture of layering on top of a conventional DBMS can be enhanced for better performance by exploiting the the uniformity in the structure of queries and encoded Trio relations. To this end, we proposed novel indexing techniques in Trio, and defined a new interesting order, which can be useful in assembling query plans. We devised a relational operator that ensures the interesting order on its output. To guide the query optimizer in choosing the optimal query plan, we designed

histograms that enable estimating various useful statistical properties of uncertain data. Finally, we suggested several avenues for interesting future work in the area of query optimization for Trio and other uncertain database systems.

References

1. S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theoretical Computer Science*, 78(1), 1991.
2. N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proc. of ACM PODS*, 1999.
3. P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE*, 2006.
4. L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing Incomplete Information with Probabilistic World-Set Decompositions. In *Proc. of ICDE*, 2007.
5. D. Barbará, H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng.*, 4(5), 1992.
6. O. Benjelloun, A. Das Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proc. of VLDB*, 2006.
7. O. Benjelloun, A. Das Sarma, A. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB Journal*, 17(2), 2008.
8. J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *Proc. of ACM SIGMOD*, 2005.
9. S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: how much is enough? In *Proc. of ACM SIGMOD*, 1998.
10. R. Cheng, S. Singh, and S. Prabhakar. U-DBMS: A database system for managing constantly-evolving data. In *Proc. of VLDB*, 2005.
11. N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *Proc. of VLDB*, 2004.
12. A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working Models for Uncertain Data. In *Proc. of ICDE*, 2006.
13. A. Das Sarma, M. Theobald, and J. Widom. Data modifications and versioning in trio. Technical report, Stanford InfoLab, 2008. Available at http://dbpubs.stanford.edu/pub/2008-5.
14. A. Das Sarma, M. Theobald, and J. Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *Proc. of ICDE*, 2008.
15. C. Estan and J. F. Naughton. End-biased samples for join cardinality estimation. In *Proc. of ICDE*, 2006.
16. N. Fuhr. A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In *Proc. of VLDB*, 1990.
17. N. Fuhr and T. Rölleke. A Probabilistic NF2 Relational Algebra for Imprecision in Databases. *Unpublished Manuscript*, 1997.
18. N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM TOIS*, 14(1), 1997.
19. H. G-Molina, J. Widom, and J. D. Ullman. *Database Systems: The Complete Book*. Prentice-Hall, 2002.
20. S. Ganguly, M. Garofalakis, and R. Rastogi. Processing data-stream join aggregates using skimmed sketches. In *Proc. of EDBT*, 2004.
21. M. Garofalakis and P. B. Gibbons. Wavelet synopses with error guarantees. In *Proc. of ACM SIGMOD*, 2002.

22. M. Garofalakis and A. Kumar. Deterministic wavelet thresholding for maximum-error metrics. In *Proc. of ACM PODS*, 2004.
23. P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proc. of VLDB*, 1997.
24. G. Grahne. Dependency Satisfaction in Databases with Incomplete Information. In *Proc. of VLDB*, 1984.
25. G. Grahne. Horn Tables - An Efficient Tool for Handling Incomplete Information in Databases. In *Proc. of ACM PODS*, 1989.
26. T. J. Green and V. Tannen. Models for incomplete and probabilistic information. In *Proc. of IIDB Workshop*, 2006.
27. T. Imielinski and W. Lipski Jr. Incomplete Information in Relational Databases. *Journal of the ACM*, 31(4), 1984.
28. S. K. Lee. An extended Relational Database Model for Uncertain and Imprecise Information. In *Proc. of VLDB*, 1992.
29. M. Mutsuzaki, M. Theobald, A. Keijer, J. Widom, P. Agrawal, O. Benjelloun, A. Das Sarma, R. Murthy, and T. Sugihara. Trio-one: Layering uncertainty and lineage on a conventional DBMS. In *Proc. of CIDR*, 2007. Demonstration description.
30. G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proc. of ACM SIGMOD*, 1984.
31. C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *Proc. of ICDE*, 2007.
32. C. Re and D. Suciu. Materialized views in probabilistic databases for information exchange and query optimization. In *VLDB*, 2007.
33. P. Sen and A. Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In *Proc. of ICDE*, 2007.
34. S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch. Indexing uncertain categorical data. In *Proc. of ICDE*, 2007.
35. J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Proc. of CIDR*, 2005.
36. J. Wijsen. Condensed representation of database repairs for consistent query answering. In *Proc. of ICDT*, 2003.

Implementing NOT EXISTS Predicates over a Probabilistic Database*

Ting-You Wang**, Christopher Re, and Dan Suciu

University of Washington

Abstract. Systems for managing uncertain data need to support queries with negated subgoals, which are typically expressed in SQL through the NOT EXISTS predicate. For example, the user of an RFID tracking system may want to find all RFID tags (people or objects) that have traveled from a point A to a point C without going through a point D. Such queries are difficult to support in a probabilistic database management system, because offending tuples do not necessarily disqualify an answer, but only decrease its probability. In this paper, we present an approach for supporting queries with NOT EXISTS in a probabilistic database management system, by leveraging the existing query processing infrastructure. Our approach is to break up the query into multiple, monotone queries, which can be evaluated in the current system, then to combine their probabilities by addition and subtraction to compute that of the original query. We will also describe how this technique was integrated with MystiQ, and how we incorporated the top-k multi-simulation and safe-plans optimizations.

1 Introduction

Probabilistic databases have been used recently in a variety of applications of uncertain data: in acquisition systems such as sensor nets or RFID deployments [5, 9, 10, 15], to manage data extracted by information extraction systems [8], to query incompletely cleaned data [1], data obtained as a result of imprecise data mappings [7], and data that has been anonymized to protect privacy [12].

The research on query processing on probabilistic databases has lead to techniques for processing select-project-join queries [4, 3], queries on data with explicit provenance [2], on Markov Network data [16], top-k queries [13], and queries with `having`-predicates [14].

An area that has not been explored so far are queries with negated subgoals, which, in SQL, can be expressed using the `NOT EXISTS` predicate. Such queries are an important piece of managing uncertain data. For example, in an RFID application a user may want to find all events when an RFID tag has traveled from point A to point C without going through a point D; in large scale information extraction systems s.a. `DBLive` [6] a user may want to find all databases

* This work was partially supported by NSF grants IIS-0513877 and IIS-0713576.

** Corresponding author: `tingyouuw@gmail.com`.

researchers that have never served on a VLDB committee (e.g. when a PC chair forming a Program Committee is searching for junior researchers that have never served on the PC before), or a PC chair may want to find all authors that have never published before in a database conference (a much needed query when selecting the *best newcomer award*).

Yet queries with negated subgoals are difficult to compute on a probabilistic database. In a standard (deterministic) database, if at least one witness tuple satisfies the negated subgoal then the answer is immediately disqualified. But in a probabilistic database such a witness does not immediately disqualify the answer, only reduces its probability. In fact, if there are many witnesses then their probabilities need to be aggregated in order to compute by how much to reduce the probability of the answer.

In this paper we describe a technique for computing SQL queries with `NOT EXISTS` predicates on a probabilistic database, which we have implemented on top of MystiQ, a probabilistic database system [11]. Our approach splits a query with a conjunction of `NOT EXISTS` as its predicate into multiple parts: a query to represent the positive results, and many queries to capture the not exists subgoals. Importantly, these queries are all simple select-from-where queries, which can be evaluated using the existing MystiQ infrastructure. Then, we combine their results and compute the probability of each output tuple by taking into account the semantics of the `NOT EXISTS` predicate. We also describe how to incorporate in this approach two optimizations present in MystiQ: top k-query evaluation and safe subplans.

Motivating application: Logging Activities The RFID Ecosystem project at the University of Washington [17] deploys about 132 antennas throughout the department's building and tracks thousands of RFID tags attached to objects or carried by people. As these tagged people/objects roam through the hallways, a database table is populated with their movements. In a perfect world, the table would contain exactly every instance a tagged person passes by an antenna. However, this is not the case. It turns out that there are many times that readings are dropped, which are called false negatives, and other times, extra readings are registered, which are identified as false positives. As a consequence, every reading recorded in the database is probabilistic. (We refer the reader to [15] for a full description of the probabilistic model.)

Fig. 1. Person is walking down the hall starting at A

For example, consider the diagram in Fig. 1 that shows the position of three antennas A, B, C (out of a few dozens), and suppose that we are interested in retrieving all timestamps when a user has walked from A to C going through B.

The antennas read at a frequency of four readings per second. False negatives are common (missed tags) and false positives are also frequent (readings from nearby antennas). Our system converts these uncertainties into probabilities, and as a consequence might record a sequence of readings like this:

$$A_1A_2B_3A_4A_5B_6B_7B_8C_9C_{10}B_{11}C_{12}\ldots A_{21}A_{22}B_{23}A_{24}D_{25}D_{26}D_{27}E_{28}E_{29}C_{30}$$

If the database were deterministic then (A_5, C_9) is the only answer: we don't consider (A_1, C_9) because there are intermediate readings at A, and we don't consider (A_{24}, C_{30}) an answer because obviously the user has taken a different route to get from A to C, other than through B. Our query is expressed by the following SQL statement which finds all pairs for readings (A, C) for which there exists no intermediate readings other than those at antenna B:

```
SELECT distinct r1.pid, r1.time, r2.time
FROM Data r1, Data r2
WHERE r1.time < r2.time AND r1.pid = <UserID> AND
      r2.pid = r1.pid AND
      r1.antenna = 'A' AND r2.antenna = 'C' AND
      NOT EXISTS (SELECT distinct *
                  FROM Data r3
                  WHERE r3.pid = r1.pid AND r3.time > r1.time
                        AND r3.time < r2.time
                        AND r3.antenna != 'B')
```

Evaluating this query on a probabilistic database poses important technical challenges. It is no longer the case that (A_5, C_9) is the single answer. For example (A_1, C_9) may also be an answer, namely when the offending readings A_2, A_4, A_5 are false positives. In fact the probability of the event (A_1, C_9) is $(1 \quad p_2)(1 \quad p_4)(1 \quad p5)$ (we denote p_2 the probability of the event A_2, etc: these values are stored in the database). Similarly, (A_{24}, C_{30}) is now also a possible answer, and its probability is that of all the offending witnesses being absent.

Paper Organization We start by describing the basic data model in Sec. 2. We describe the main idea in Sec. 3, then provide the general approach in Sec. 4. We describe some implementation details in Sec. 5 and some preliminary experiments in Sec. 6. We conclude in Sec. 7.

2 Data and Query Model

We briefly describe MystiQ's query and data model from [3]. The relations are independent/disjoint probabilisitic relations, in which every two tuples are either independent, or disjoint events. Queries are expressed in SQL, and each answer is annotated by a probability representing the system's confidence in that answer; the answers are typically presented to the user in decreasing order of their output probability, see Fig. 2. MystiQ supports SELECT-DISTINCT-FROM-WHERE

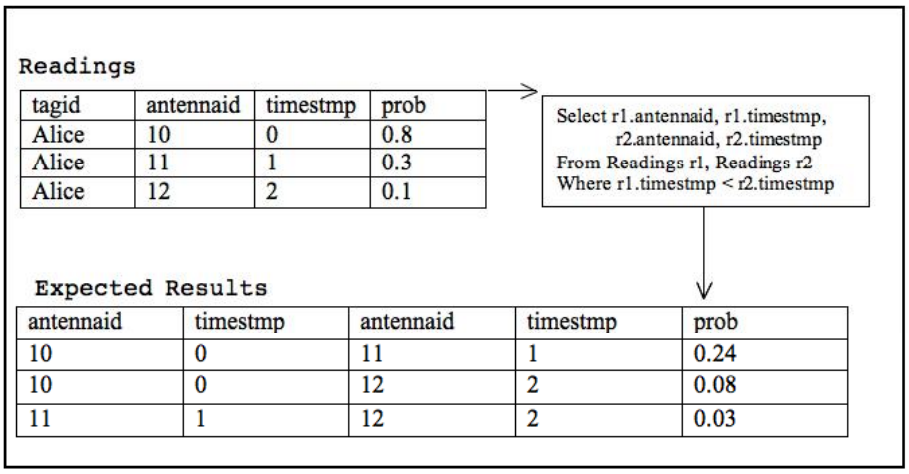


Fig. 2. An example of a query for MystiQ

queries and uses two algorithms to compute probabilities: *safe plans* [4] and *multisimulation* [13].

The first algorithm translates a SQL query Q into another query Q' that manipulates probabilities explicitly: it multiplies or adds them, or computes their complement $(1-p)$. When such a rewriting is possible then Q is called a *safe query* and Q' is called a *safe plan*. The relational database server executes Q' and returns tuples ordered decreasingly by their probabilities: MystiQ will display them without any further processing.

The second algorithm is used when Q is not safe. In this case MystiQ runs a much simpler query on the database engine, but needs to do considerably more work to compute the probabilities. The database engine simply returns all possible answer tuples $t_1, \ldots, t_n$ together with their *lineage* [2]: it does not compute any output probabilities. The lineage of a tuple is a positive DNF formula whose propositional variables are tuples from the input database: importantly, these input tuples carry with them the input probability. The probability of each tuple t_i is precisely the probability of its associated DNF formula, and MystiQ evaluates these probabilities p_1, p_2, ..., p_n by running n Monte Carlo simulation algorithms. This is an expensive process, and here MystiQ uses an optimization called *multisimulation*, introduced in [13], whose goal is to run the smallest number of simulation steps required to identify and rank only the top k tuples, ordered decreasingly by their probabilities. We describe here how the multisimulation algorithm computes the top k tuples, without necessarily sorting them, and refer the reader to [13] for details. Suppose that at some point we have a confidence interval for each tuple: $[a_1, b_1]$, $[a_2, b_2]$, ..., $[a_n, b_n]$, s.t. it is guaranteed that $p_i \in [a_i, b_i]$, for $i = 1, \ldots, n$. (Initially, $a_1 = a_2 = \ldots = 0$ and $b_1 = b_2 = \ldots = 1$.) The crux of the algorithm consists of choosing which DNF

formula to simulate next. Let c be the k'th largest value in $\{a_1, \ldots, a_n\}$ and let d be the $k+1$'st largest value in $\{b_1, \ldots, b_n\}$. The first observation is that if $d < c$ then the algorithm has succeeded in identifying the top k tuples: these are precisely the k tuples t_i s.t. $c \leq a_i$ since for any other tuple t_j not in this set it is the case that $b_j \leq d$, hence $p_j < p_i$. If $c \leq d$, then the interval $[c, d]$ is called *the critical region*: the second observation is that in this case one must improve the confidence interval $[a_i, b_i]$ of some tuple that contains the critical region: $a_i \leq c < d \leq b_i$. Thus, the algorithm chooses one of the tuples t_i whose current confidence interval $[a_i, b_i]$ "crosses" the critical region (hence t_i is called a *crosser*), then runs a few simulation steps on the DNF formula for t_i, which further shrinks the interval $[a_i, b_i]$ and then re-computes the critical region c, d. It stops when $d < c$.

In summary, MystiQ tries to evaluate a safe query, when possible, or runs the multisimulation algorithm otherwise.

3 Evaluating Queries with a Single NOT EXISTS

In this work we considered SQL queries that have a conjunction of NOT EXISTS in the `where` clause; the nested queries are in turn simple `select-from-where` queries, i.e. we do not allow nested NOT EXISTS. We describe in this section our approach, and we start with the simple case of a single nested NOT EXISTS query. We use the following as our running example:

```
Original query Q:
    SELECT distinct col_1, col_2, col_3
    FROM R1, R2, R3
    WHERE condition1 AND NOT EXISTS
        ( SELECT *
          FROM R'1, R'2, R'3
          WHERE condition2 )
```

From Q, we derive the following two queries. The *outer query* is the query Q with the NOT EXISTS predicate removed:

```
Outer query Q1:
    SELECT distinct col_1, col_2, col_3
    FROM R1, R2, R3
    WHERE condition1
```

The *inner query* is the query under the NOT EXISTS predicate:

```
Inner query Q2:
   SELECT distinct col_1, col_2, col_3
   FROM R1, R2, R3, R'1, R'2, R'3
   WHERE condition1 AND condition2
```

Let E_1 be the event that a tuple t is in the answer to $Q1$, and $E2$ be the event that t is in the answer to $Q2$. Then the answers to Q are precisely the tuples satisfying the event $E_1\overline{E_2}$ and their probability is:

$$P(t \in Q) = P(E_1) - P(E_1E_2) = P(E_1) - P(E_2)$$

The last equality holds because every tuple that is an answer to Q_2 is also an answer to Q_1.

In summary, to evaluate Q on a probabilistic database we proceed as follows. First we evaluate $Q1$ and obtain a set of tuples $t_1, \ldots, t_n$ with probabilities $p_1, \ldots, p_n$. Next, we evaluate $Q2$: we assume its answer is the same list of tuples, with probabilities $p'_1, \ldots, p'_n$: if $Q2$ returns some tuple that is not among $t_1, \ldots, t_n$ then we can ignore it, and conversely, if it does not return some tuple t_i we simply add it and set its probability $p'_i = 0$. The answer to the query Q is the list of tuples $t_1, \ldots, t_n$, and their probabilities are $p_1 \quad p'_1, \ldots, p_n \quad p'_n$.

4 Evaluating Queries with Multiple NOT EXISTS

We now show how to generalize to multiple NOT EXISTS predicates. Our running example is:

```
Q:
SELECT DISTINCT col_1, col_2, col_3
FROM A_1, B_1, C_1
WHERE condition_1 AND NOT EXISTS
   (SELECT *
    FROM A_2, B_2, C_2
    WHERE condition_2)
    AND
    ...
    AND NOT EXISTS
   (SELECT *
    FROM A_m, B_m,C_m
    WHERE condition_m)
```

As before we define an *outer query* $Q1$, and several *inner queries* $Q2$, $Q3$, ..., Qm; their definitions are by a straightforward generalization of those in the previous section.

Let E_i be the event that a tuple t is in the answer set to the query Qi. Then, the event "t is in the answer to Q" is equivalent to:

$$(t \in Q) = E_1\overline{E_2}...\overline{E_k}...\overline{E_m}$$

Therefore, by using the inclusion-exclusion formula we derive:

$$
\begin{aligned}
P(t \in Q) &= P(E_1\overline{E_2}...\overline{E_k}...\overline{E_m}) \\
&= P(E_1) \quad P(E_1(E_2 \vee E_3 \vee \ldots \vee E_m)) \\
&= P(E_1) - \sum_{2 \leq i_2 \leq m} P(E_1E_{i_2}) + \sum_{2 \leq i_2 < i_3 \leq m} P(E_1E_{i_2}E_{i_3}) - \ldots - (-1)^m P(E_1E_2 \ldots E_m)
\end{aligned}
$$

Thus, in order to evaluate the query Q we proceed as follows. We start by evaluating the query Q_1: suppose it returns n tuples, $t_1, \ldots, t_n$, with probabilities $p_1, \ldots, p_n$. Next we evaluate the following $2^{n-1} \quad 1$ queries: $Q_{i_2} \ldots Q_{i_k}$ for $k \geq 1$ and $1 \leq i_2 < \ldots < i_k \leq n$. (Note that $Q_1Q_{i_2} \ldots Q_{i_k}$ is equivalent to $Q_{i_2} \ldots Q_{i_k}$ when $k > 0$.) Assume that each of these queries returns the same set of tuples $t_1, \ldots, t_n$, with some probabilities. (Any additional tuples can be removed, and any missing tuples can be added with probability 0.) Then the answer to Q consists of the tuples $t_1, \ldots, t_n$, and the probability of t_i is obtained by summing the $2^n \quad 1$ probabilities of t_i in all queries $Q_1Q_{i_2} \ldots Q_{i_k}$, with signs $(-1)^{k+1}$.

We end this section with a note on our algorithm for generating a list of $2^n \quad 1$ queries that need to be evaluated. This process is briefly demonstrated in Fig. 3. We maintain a list of already generated queries and then appending onto the list by merging new nested queries with those already in the list. However, the order of the appending is very important; it must be from the back of the list to the front in order to create the alternating property that we desire. With this we are ready to begin the implementation process for NOT EXISTS.

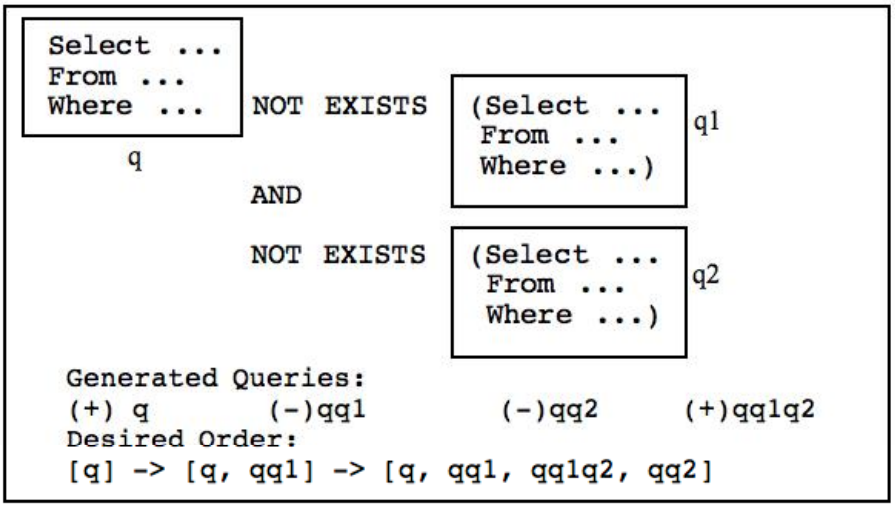


Fig. 3. Example of how a list of generated queries is populated

5 Implementing NOT EXISTS

It is clear that the first step of implementing NOT EXISTS is to parse a query and recognize the NOT EXISTS clauses. Finding the clauses is straightforward since it is a standard traversal of a tree generated to represent the input query. But once these locations are found, we cannot immediately begin the generation of queries because there are two main issues that must be resolved. First, the naming of table aliases could conflict between the outer and inner queries. Second, we need to generate the queries in the correct order to produce the alternating in signs that we desire for the probabilities.

The first problem requires a renaming that is done at the moment an input query is parsed. For every table alias that is given, we will append on a suffix to make the name unique from all others. The suffix chosen for MystiQ was _X where X is a non-negative integer. An example of this renaming is shown below:

```
SELECT R1.attr1, R1.attr2, R2.attr1
FROM Relation R1, Relation R2
WHERE R1.attr1 = R2.attr2 and R1.attr2 = 'in' AND NOT EXISTS
   (SELECT *
    FROM Relation R1
    WHERE R1.attr1 = R2.attr2 and R1.attr2 = 'out')

TRANSLATES INTO:

SELECT R1_1.attr1, R1_1.attr2, R2_2.attr1
FROM Relation R1_1, Relation R2_2
WHERE R1_1.attr1 = R2_2.attr2 and R1_1.attr2 = 'in' and NOT EXISTS
  (SELECT *
   FROM Relation R1_3
   WHERE R1_3.attr1 = R2_2.attr2 and R1_3.attr2 = 'out')
```

The second problem is much more simple to solve. In order to generate the new queries in the correct order, we first make a pass over the input query and extract/remove all the inner queries. At the end of this process, we have a list of all the nested queries and the outer query we need. For the first of the nested queries, we merge it with the outer query, then we take another nested query, if it exists, and merge it with the first two generated queries in reverse order (just as Fig 3 demonstrated). We proceed in this fashion for all nested queries. After this process is done, we are ready to begin calculating probabilities.

Calculation of probabilities has two pathways that it can take. The first is to compute the probabilities explicitly, taking advantage of the safe plans optimization. The second is to use the multi-simulation method. Clearly, explicitly calculating the probability is much more desirable, so unless a user specifies that simulations must be run, MystiQ will check if all of the generated queries, which are all monotone, can be computed exactly. If so, we can compute the NOT EXISTS query by substituting the probability values directly into the formulas discussed in Sections 3 and 4 to get an exact solution.

In a simulation, we no longer have exact probabilities to work with, but rather entire sets of confidence intervals, one set for each of the generated queries. Recall from Section 2, an *interval* represents a range in which the true probability lies for a particular result of the query. As mentioned in Section 4, the possible tuples of the original query are precisely those of the outer query. However, there is the additional complication of the dependency between the intervals for the original query and those of the generated queries. Since we never actually ran the original query we have no DNF formula to simulate, so we must use the other intervals.

The endpoints for a tuple of the original query is equivalent to just finding the extreme points of the probability formula. For example, if one was computing the lower endpoint of an interval, they would be computing the smallest possible probability value that could result from substituting the probabilities of the equivalent tuples of the generated queries. Similarly, the upper endpoint would be the largest possible probability value. Figure 4 provides a demonstration of the actual calculation and equations.

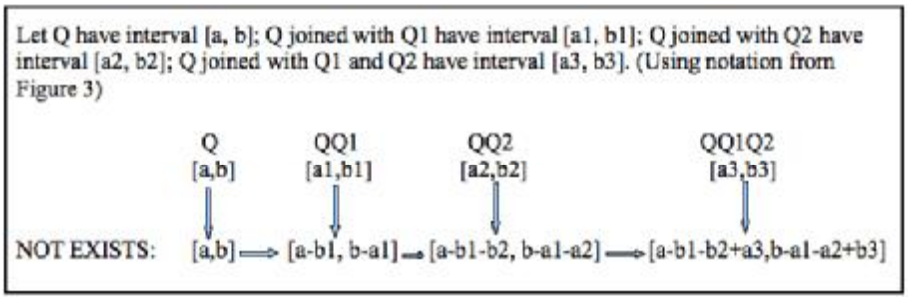


Fig. 4. Example of how intervals are calculated

With this key issue resolved, the simulation process can proceed in much the same way as it did in the original implementation of MystiQ, with only a slight modification. The process begins with initializing a set of intervals (to represent the original query), S, to the intervals of the outer query. Then, one interval, $I \in S$, is chosen (exactly in the same way an interval would have been chosen before). However, this interval itself is not simulated. Rather, all intervals representing the same tuple from the generated queries are simulated. These associated intervals are then aggregated together using the method just described before to come up with the new interval for our original query. This process is repeated until a certain number of intervals are isolated from each other giving us the top-k tuples.

6 Experiments

Experiments were run in order to judge how well the implementation performed when adjusting various parameters. For starters, we adjusted database table sizes and toggled using safe plans. Next, we varied the number of Monte Carlo simulations run every step in the multisimulation algorithm. Finally, we experimented with how the query system handles more complex queries.

First, experiments were run to test how the implementation performed over different data sizes. For this, we used a synthetic database (rows are generated with 'random' probabilities) consisting of `Products` and `Orders`, and ran the following query:

```
SELECT DISTINCT p.id, p.name
FROM ProductEvent p
WHERE NOT EXISTS(
          SELECT DISTINCT *
          FROM OrderEvent o
          WHERE o.productid = p.id and o.price > 10)
```

We varied the data set size (split almost equally between `Products` and `Orders`): ∼4000, ∼9500, ∼14000, ∼19000, ∼28500 rows. We also ran this query, both, with the safe plan optimizations and without. Fig. 5 shows the results of those runs.

Fig. 5. Comparison of optimized and unoptimized queries

From this graph, we see that the running times of NOT EXISTS queries exponentially rise as the data set size increases. However, as the size increases, the slopes do seem to grow shallower. We also see in the graph that safe plan optimizations make a significant improvement in performance. From the tests,

there was generally a 94% improvement in time when optimizations could be found.

Another experiment examined how changing the number of simulations run every step in the simulating process of MystiQ affected the running time. The graph in Fig. 6 charts the change in times as we varied the number of simulations from 10000 to 70000 on a fixed dataset of about 10000 rows, about 5000 products and 5000 orders (we used the same query as before).

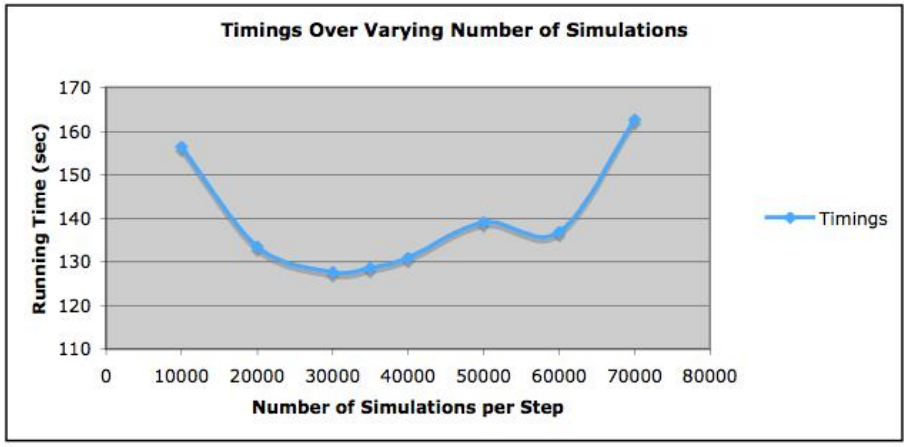


Fig. 6. Execution times against changing the number of simulations per step

We see that for both small and large values, the running times dramatically increase; however, the cause is very different. For the smaller values, the increase in running time is mostly due to the fact that many more steps have to run to reach enough simulations on the intervals to isolate the top k. For the larger values, the increase in time is due to the fact that, though we reduce the number of steps run, we are over simulating the intervals. The extra simulations are simply taking up time and not adding any new information. The middle numbers, particularly at around 30000, balance the two problems well. They run enough simulations to reduce the number of steps taken, but they do not over simulate the intervals.

Although Fig. 6 showed 30000 to be the optimal value for the `Number of Simulations per Step`, this is only for this particular case. Different queries will have different results. However, even in those cases, the graph of times ought to follow the same curve since these queries face the same problems, only it will be translated along the x-axis.

Lastly, experiments were performed on gathered data closely related to the RFID data (times when a person entered and left a room). A complex query was written that searched for consecutive events of entering and leaving a room

for a particular person, but the data set size was only 660 tuples (split between entering and leaving rooms). The query, at first, used two NOT EXISTS queries. Here is an example of the query:

```
SELECT DISTINCT er.tag_id, er.room_num, er.timestamp, lr.timestamp
FROM EnteredRoomEvent er, LeftRoomEvent lr
WHERE er.tag_id = lr.tag_id and er.room_num =
       lr.room_num and er.timestamp < lr.timestamp AND NOT EXISTS
   (SELECT DISTINCT *
    FROM EnteredRoomEvent e3
    WHERE e3.tag_id = er.tag_id and e3.timestamp >
          er.timestamp and e3.timestamp < lr.timestamp) AND NOT EXISTS
   (SELECT DISTINCT *
    FROM LeftRoomEvent e4
    WHERE e4.tag_id = er.tag_id and e4.timestamp >
         er.timestamp and e4.timestamp < lr.timestamp)
```

Then the query was translated into a single NOT EXISTS query to see how timings are affected. This new query is found below (where the AllSightings table is simply the union of EnteredRoom/LeftRoom events).

```
SELECT DISTINCT er.tag_id, er.room_num, er.timestamp, lr.timestamp
FROM EnteredRoomEvent er, LeftRoomEvent lr
WHERE er.tag_id = lr.tag_id and er.room_num =
       lr.room_num and er.timestamp < lr.timestamp AND NOT EXISTS
      (SELECT distinct *
       FROM AllSightings all
       WHERE all.tag_id = er.tag_id and all.timestamp >
                      er.timestamp and all.timestamp < lr.timestamp)
```

The results of the test showed that it is about 25 times slower when adding one additional level of complexity. With only one NOT EXISTS sub query, results were found in 51.062 seconds where as by adding one more level of complexity, it took 1260.389 seconds. This increase comes from executing twice the number queries and running simulations on twice the number of sets of intervals.

In addition to the immediate results of the experiment, another observation was made. An issue arises when there are many probabilities that lie very close together, particularly when they are all close to zero (which was the case for this experiment). While executions were run, there was a clear slow down when all the intervals narrowed and were centered about zero. At this point, the simulator makes only small changes to an interval's endpoints leading to longer execution to distinguish a top k set of intervals. This is more prevalent in NOT EXISTS queries since using the NOT EXISTS clause, we can more easily send more probabilities closer to zero, but if certain monotone queries were chosen carefully over data sets with very small probabilities, it would also be possible for this case to arise.

7 Conclusion

Systems for managing uncertain data need to support queries with negated subgoals, such as SQL queries with NOT EXISTS predicates. We have described in this paper an approach for supporting such queries, while leveraging much of the infrastructure of an existing probabilistic database management system. We have described optimizations, which in our preliminary experiments show improvement of performance by about 94%. However, the execution time still is fairly slow and can be improved upon. As our experiments show, the running times are seemingly exponential with respect to the data size. Also, as the complexity of the query increases, there is also a significant increase in running time. Further improvements in any of these areas would be extremely beneficial to making the system more practical to use in many different environments.

Possible directions for future work would be to improve the simulation process or improving the interpretation of a NOT EXISTS query. Some specific areas that can be improved include implementing a faster way of running Monte Carlo steps, or understanding how to better choose intervals to simulate, or even figuring out how to choose the number of steps to take each time we simulate an interval. Other work could be to find better ways to break apart a NOT EXISTS query so that we no longer have an exponential number of queries. Unless we get away from the exponential increase in generated queries, we can always expect that there will probably be an exponential increase in time when more NOT EXISTS queries are added.

References

1. P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases. In *ICDE*, 2006.
2. O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widom. An introduction to ULDBs and the Trio system. *IEEE Data Eng. Bull*, 29(1):5–16, 2006.
3. N. Dalvi, C. Re, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1):25–31, 2006.
4. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, Toronto, Canada, 2004.
5. A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Using probabilistic models for data management in acquisitional environments. In *CIDR*, pages 317–328, 2005.
6. A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. *IEEE Data Engineering Bulletin, Special Issue on Probabilistic Data Management*, 29(1):64–72, March 2006.
7. X. Dong, A. Halevy, and C. Yu. Data integration with uncertainty. In *VLDB*, 2007.
8. R. Gupta and S. Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, pages 965–976, 2006.
9. T. Jayram, S. Kale, and E. Vee. Efficient aggregation algorithms for probabilistic data. In *SODA*, 2007.

10. S. Jeffery, M. Garofalakis, and M. Franklin. Adaptive cleaning for RFID data streams. In *VLDB*, pages 163–174, 2006.
11. MystiQ: a probabilistic database system, avialable at `http://mystiq.cs.washington.edu/`.
12. V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. In *VLDB*, 2007.
13. C. Ré, N. Dalvi, and D. Suciu. Efficient Top-k query evaluation on probabilistic data. In *ICDE*, 2007.
14. C. Re and D.Suciu. Efficient evaluation of having queries on a probabilistic database. In *Proceedings of DBPL*, 2007.
15. C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Extracting events from correlated streams. In *SIGMOD*, page *to appear*, 2008.
16. P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, 2007.
17. E. e. a. Welbourne. Challenges for pervasive RFID-based infrastructures. In *PERTEC Workshop*, March 2007.

Indexing Probabilistic Nearest-Neighbor Threshold Queries

Yinian Qi[1], Sarvjeet Singh[1], Rahul Shah[2], and Sunil Prabhakar[1]

[1] Department of Computer Science, Purdue University
{yqi, sarvjeet, sunil}@cs.purdue.edu
[2] Department of Computer Science, Louisiana State University
rahul@csc.lsu.edu

Abstract. Data uncertainty is inherent in many applications, including sensor networks, scientific data management, data integration, location-based applications, etc. One of common queries for uncertain data is the probabilistic nearest neighbor (PNN) query that returns all uncertain objects with non-zero probabilities to be NN. In this paper we study the PNN query with a probability threshold (PNNT), which returns all objects with the NN probability greater than the threshold. Our PNNT query removes the assumption in all previous papers that the probability of an uncertain object always adds up to 1, i.e., we consider missing probabilities. We propose an augmented R-tree index with additional probabilistic information to facilitate pruning as well as global data structures for maintaining the current pruning status. We present our algorithm for efficiently answering PNNT queries and perform experiments to show that our algorithm significantly reduces the number of objects that need to be further evaluated as NN candidates.

1 Introduction

The nearest neighbor (NN) query is one of the most common database queries that finds the nearest object to a query object, given some distance function. Many algorithms have been proposed for NN queries [13, 3, 11] where the value of data is certain. However, uncertainty in data is inherent in many applications, such as sensor networks and location-based applications [5], where data can take different values with probabilities due to measurement errors. Take the location-based data for example. Suppose all data objects are in 2-dimensional space. The exact location of an object is unknown. However, each object is associated with a region of its possible locations and the probability density function (pdf) of the object's location within the region is known. In this probabilistic data setting, since each object has a probability (maybe 0) to be NN to a query object, we have to take probabilities into account when answering NN queries: We can either return all objects with a non-zero probability to be NN or return all objects with the NN probability greater than some threshold. We call the former *probabilistic nearest neighbor (PNN) queries* and the latter *probabilistic nearest neighbor threshold (PNNT) queries*. The formal definitions of both

queries are presented in Section 1.1. Several papers have studied the NN problem with uncertain data. For example, [5] proposed an algorithm for answering PNN queries. The algorithm returns all objects along with their non-zero NN probabilities, which requires a large number of expensive computations of the exact NN probabilities. However, most of the time we are only interested in objects with a relatively large probability to be NN, hence a probability threshold can be specified for the query to only return objects with NN probability that meets the threshold (PNNT queries). For such queries, the threshold can be leveraged to prune objects that cannot satisfy the probability requirement. [8] proposed the constrained probabilistic nearest neighbor query (C-PNN) with both threshold (P) and tolerance (Δ) constraints, which is equivalent to our PNNT query with the threshold being $P \quad \Delta$.

We generalize the above NN problems for uncertain data by taking into account missing probabilities, which is not addressed in any of the previous NN papers such as [5, 8]. The missing probabilities of an object result in its absence. For the location-based data, this means that the probability of the object's location in its uncertain region may not add up to 1, indicating that there may be some probability that the object does not exist. In an uncertain database, if we consider each object to be a tuple in a relation, then the uncertain data in our NN problem has both attribute and tuple uncertainty. This is consistent with the recent uncertain model [16] where both the value of a given attribute in a tuple and the presence of the tuple itself may be uncertain. For attribute uncertainty, we assume that the uncertain attribute is associated with a pdf. For tuple uncertainty, the probability of its presence is the sum of probabilities over the pdf of all its attributes, which can be less than 1.

In this paper, we limit our discussion to NN queries with at most two uncertain attributes (i.e., 2-dimensional space) for simplicity, although our approach does not have such restriction and can be extended to higher-dimensional space. We now formally define our NN problem for uncertain data, and show why it is more complicated with missing probabilities.

1.1 Problem Definitions

Assuming a database of objects with uncertain attributes as continuous random variables associated with pdfs, we give two formal problem definitions for NN queries of such objects.

Definition 1. ***Probabilistic Nearest Neighbor (PNN) Query****: Given a query point q and a set of objects with uncertain attributes and their corresponding pdfs, a PNN query returns the probability $P_{nn}(O)$ that object O is NN to q for each object O.*

For PNN queries, the probability for each object to be NN must be computed unless there is evidence that the object cannot be NN (i.e., the NN probability is 0). This implies a huge number of computations if the number of objects is huge. Moreover, the computation of the probability itself is very expensive,

which depends on many other objects whose uncertain regions overlap with its own [5]. The exact probability computation can involve integrations over multiple subregions that may have arbitrary pdfs, resulting in a high computational cost. However, objects having a small probability to be NN are generally less important than those with a high probability. For many applications, it is only necessary to retrieve objects with the NN probability exceeding a given threshold. The formal definition of such queries is given below.

Definition 2. ***Probabilistic Nearest Neighbor Threshold (PNNT) Query:*** *Given a query point q, a threshold τ and a set of objects with uncertain attributes and their pdfs, the PNNT query returns every object O with $P_{nn}(O) \geq \tau$.*

Since we are only concerned about object O with $P_{nn}(O) \geq \tau$ in PNNT queries, we do not need to compute the exact P_{nn} of the object if we can prove the probability cannot exceed τ. In this case, we can safely prune away those objects, hence reduce the computational cost.

Note that in our PNNT queries, we do not require that the probabilities of an object's region sum up to 1 (in other words, the pdf can be a partial pdf). Suppose the sum is p, then $1-p$ is the missing probability that the object does not exist at all. This is a more general case. We need to consider more when pruning objects: Unless at least one object closer to q is sure to exist, an object that is far from q still has a non-zero probability to be NN, thus cannot be pruned away immediately as in [8].

The main contributions of our paper consist of the following: We proposed an R-tree based index structure that can efficiently answer PNNT queries, which is a normal R-tree index augmented with additional probability information so that we can prune objects away without sacrificing the correctness of the results. Furthermore, we designed global data structures to maintain and update the current pruning status as we retrieve nodes from the R-tree. Our PNNT query processing algorithm overcomes the limitation of previous NN papers with regard to missing probabilities and is proven to be efficient by our experiments.

The rest of the paper is organized as follows: Section 2 introduces probabilistic information that will be added to the R-tree index. Section 3 presents the algorithms used in PNNT query processing, followed by the experimental results in Section 4. Section 5 reviews the related work. Finally, we conclude our paper and point out future work in Section 6.

2 Augmented R-tree Index

In this section, we describe our new R-tree index for the PNNT problem defined in the previous section. We propose three types of augmentation to the normal R-tree in order to answer the PNNT queries both effectively and efficiently. The following information is added to the entries in an R-tree to facilitate query processing: Absence probability (AP), maximal probability (MP) and the absence probability bounds (AP-bounds). We first introduce each augmentation separately, then show how to incorporate all of them into our index structure. In the rest of the paper, we use τ to denote the PNNT query threshold.

2.1 Absence Probability (AP)

Definition 3. ***Pruning Circle****: A circle $C_{q,r}$ centered at query point q with a radius r is called a* pruning circle *if for every object O lying outside $C_{q,r}$ we have $P_{nn}(O) \leq \tau$.*

The reason why $C_{q,r}$ is called a pruning circle is that given $C_{q,r}$, we can safely prune away all objects lying outside it when processing PNNT queries. Our goal is to shrink the pruning circle as much as possible so that all objects outside it can be pruned away immediately, leaving only a small portion of objects to be further examined as NN candidates. Next we introduce absence probability for our augmented R-tree index:

Definition 4. ***Absence Probability*** *AP: Given a Minimum Bounding Rectangle (MBR) M in an R-tree, $AP(M)$ is defined as the probability that none of the objects contained in M is present. Likewise, for a circle C, $AP(C)$ is the probability that no object in C is present.*

Moreover, we define *maximum distance* $d_{max}(q, M)$ from query point q to MBR M to be the maximum distance of all distances from q to M and similarly *minimum distance* $d_{min}(q, M)$ is the minimum distance of all distances from q to M. We propose the following theorem that leverages $AP(M)$ and $d_{max}(q, M)$ to prune away MBRs whose objects cannot be NN candidates.

Theorem 1. *If $AP(M_i) \leq \tau$ for MBR M_i, then a circle $C_{q,r}$ centered at query point q with radius $r = d_{max}(q, M_i)$ is a pruning circle.*

Proof. Since there may be objects inside $C_{q,r}$ that are contained in MBRs other than M_i (denoted as M_j, as shown in Fig. 1), we can infer that

$$AP(C_{q,r}) \leq AP(M_i) \cdot \left(\prod_{M_j, j \neq i} AP(M_j) \right) \leq AP(M_i) \leq \tau$$

For any object O in any MBR M_k outside $C_{q,r}$ ($d_{min}(q, M_k) \geq r$) to be NN to q, there should be no object inside $C_{q,r}$, i.e., $P_{nn}(O) \leq AP(C_{q,r}) \leq \tau$. From Definition 3 we conclude that $C_{q,r}$ is a pruning circle.

Fig. 1 illustrates the pruning circle $C_{q,r}$ when $AP(M_i) \leq \tau$. The MBR M_k outside the circle thus can be pruned away immediately. This pruning strategy with respect to AP will be referenced later as the ***first-level pruning***. We will see in Section 2.3 a variation of it that is finer grained.

2.2 Maximal Probability (MP)

Definition 5. ***Maximal Probability*** *$MP(M)$ for MBR M is defined as $\max_{O \in M} p_o$, where O is an object contained in M and p_o is the probability that O is present.*

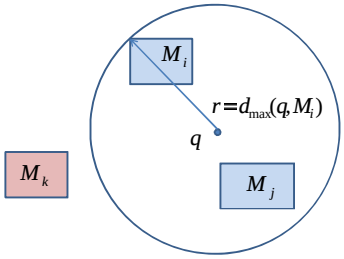


Fig. 1. Pruning Circle $C_{q,r}$ ($AP(M_i) < \tau$)

The maximal probability MP is introduced for two purposes. Firstly, it can be used for ***pre-pruning*** to prune away MBRs with $MP < \tau$. Consider an MBR M with $MP(M) < \tau$. By definitions of MP and p_o, we know that $p_o \leq MP(M) < \tau$. Since the probability for O to be NN to q is at most the probability of its presence, we have $P_{nn}(O) \leq p_o < \tau$ for any object O in M. Hence we can safely prune away the entire M. Secondly, $MP(M)$ can also be used for further pruning beyond the capability of the first-level pruning, which we call the ***second-level pruning***, which is supported by the theorem below:

Theorem 2. *Let M_i be an MBR within a circle C. Let M_k be an MBR outside C. If $MP(M_k) \cdot AP(M_i) < \tau$, then for any object O in M_k, $P_{nn}(O) < \tau$.*

Proof. For any object O in M_k, we have $P_{nn}(O) \leq p_o \cdot AP(C) \leq MP(M_k) \cdot AP(M_i) < \tau$, where p_o is the probability that O is present.

We have proved above the probability that any object O in M_k is NN to q is less than τ, so we can safely prune away the entire MBR M_k. This is called *second-level pruning*. In Fig. 1, if $AP(M_i) \geq \tau$ instead, we cannot use the first-level pruning to prune away M_k. However, if $MP(M_k) \cdot AP(M_i) < \tau$ holds, we can use the second-level pruning to prune M_k away.

2.3 *AP*-Bounds

Unlike AP introduced in Section 2.1 that stores the absence probability of an entire MBR, AP-bounds store the absence probabilities of regions in the MBR specified by the bounds. The goal of AP-bounds is to shrink the size of the pruning circle as much as possible so that more MBRs outside the circle can be pruned away. This is a fine grained version of the first level pruning in Section 2.1. Both methods require that we have a pruning circle in which the absence probability is below τ.

The idea of probability bounds (e.g. x-bounds) is first proposed in [6] for range queries with probability thresholds. In our paper, however, we use probability bounds for PNNT queries.

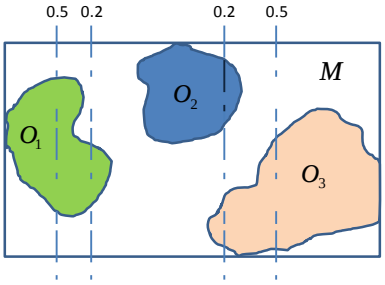


Fig. 2. *AP*-bounds in MBR M

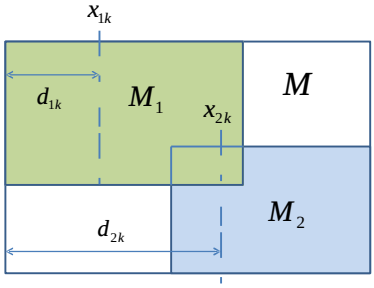


Fig. 3. Construct left-*AP*-bounds of M

Definition 6. *AP-bounds* $AP^l_M(x)$ *(left AP-bound) and* $AP^r_M(x)$ *(right AP-bound) for MBR* M *are defined as a pair of lines intersecting with* M *such that the absence probability of the region to the left of* $AP^l_M(x)$ *and to the right of* $AP^r_M(x)$ *is no greater than the bounding probability* x *(*$0 \leq x \leq 1$*).*

Furthermore, we define *AP-distances* $d^l_M(x)$ and $d^r_M(x)$ to be distances from the left and right edges of MBR M to $AP^l_M(x)$ and $AP^r_M(x)$ respectively. We require that *AP*-bounds be tight — they are pushed towards the left or right edges of the MBR as much as possible while still satisfying Definition 6. This ensures that *AP*-bounds are unique. *AP*-bounds can be represented using *AP*-distances and the bounding probability x. For example, $AP^l_M(x)$ is represented using distance $d^l_M(x)$ and x itself. Suppose O_1, O_2, and O_3 are three objects in MBR M, as shown in Fig. 2. Let the bounding probability be x, then the *AP*-bounds of M ensures that the probability that none of the three objects is present within the *AP*-bounds is no more than x. Note that the bounding probability becomes larger as *AP*-bounds are pushed towards the edges, i.e., $0.5 > 0.2$ in Fig. 2.

Pruning with *AP*-bounds: We first find the set of *AP*-bounds with bounding probability $x < \tau$ and as close to the edges of the MBR as possible. We let the new radius of the pruning circle be the minimal distance from the query point q to the *AP*-bound. Then all MBRs outside of this circle can be pruned away. Pruning using *AP*-bounds instead of *AP* of an entire MBR has the advantage that the resulting pruning radius is smaller, indicating that more MBRs are likely to be outside of the pruning circle and thus can be discarded immediately without further evaluation.

2.4 The Index Structure

Now that we have introduced all three kinds of information that we want to leverage in PNNT query processing, we redesign the R-tree index structure by adding all the information to the entries of the R-tree internal nodes. The construction of the augmented R-tree index is also discussed in details.

There are multiple entries in an R-tree internal node, each of which has an MBR (M) and a pointer (p) to a child node that stores information about all smaller MBRs contained in M. We augment R-tree by adding the following additional items to each entry of an internal node: i) AP ii) MP iii) left AP bounds and right AP bounds. Note that we keep a set of left and right AP bounds with different bounding probabilities x to suit queries of various thresholds. The list of x's is stored globally, each of which corresponds to a left and right pair of AP bounds in the MBR entry.

When constructing the new index, we propagate the additional information in MBR entries in a bottom-up fashion: The AP of an MBR at a higher level of the R-tree can be obtained by simply multiplying APs of all its child MBRs. Let $M_1, M_2, ..., M_m$ be the child MBRs of M. Then $AP(M) = \prod_{k=1}^{m} AP(M_k)$. In contrast, the MP of MBR M is obtained by finding the maximum MP among its child MBRs, i.e., $MP(M) = \max_{k=1}^{m} MP(M_k)$. To compute the left-$AP$-bounds $AP^l_M(x)$ of M, we compute the AP distance $d^l_M(x) = \max_{k=1}^{m} d^l_{M_k}(x)$ for each bounding probability x. The right-AP-bounds are computed in the same way. We call this method "Coarse Estimation Method" (CEM). Alternatively, we have "Fine Estimation Method" (FEM), which leverages the *AP hop function* to obtain a much finer estimation of AP-bounds. We compute AP hop functions for all of M's child MBRs and deduce the hop function of M from them.

Definition 7. ***AP hop function*** *is a function from AP-distance d to bounding probability x, denoted as $x = h(d)$. A hop function is with regard to an MBR M if d is the distance from AP-bounds to M's bounds.*

Note that for both left and right AP-bounds, we have a corresponding hop function. Suppose M_1 and M_2 are two MBRs contained in M, as shown in Fig. 3. $x_{11} \cdots x_{1m}$ are the bounding probabilities of left AP-bounds ($AP^l_{M_1}$) of M_1. Likewise, $x_{21} \cdots x_{2m}$ are the bounding probabilities of left AP-bounds ($AP^l_{M_2}$) of M_2. Let h_1 be the hop function of M_1 and h_2 for M_2. Let (d_{jk}, x_{jk}) be the points on h_j, where $j \in \{1, 2\}$, $1 \leq k \leq m$, and d_{jk} is the distance from M's left edge to M_j's AP-bound $AP^l_{M_j}(x_{jk})$. Moreover, the AP-bounds for both MBRs are ordered such that $d_{jk} < d_{jk+1} (d_{jm+1} = +\infty, d_{j0} = 0)$. Then we write function h_j as follows:

$$h_j(d) = x_{jk}, \quad \text{if} \quad d_{jk} \leq d < d_{jk+1} \tag{1}$$

Our goal is to compute M's hop function h from h_1 and h_2. The absence probability of the region within the AP-bound $AP^l_M(x)$ with AP-distance at least $\max(d_{1k_1}, d_{2k_2})$ is at most the product of absence probabilities within AP-bounds $AP^l_{M_1}(x_{1k_1})$ and $AP^l_{M_2}(x_{2k_2})$, that is, AP^l_M's bounding probability $x \leq x_{1k_1} \cdot x_{2k_2}$.

Having observed this property, we can obtain h from h_1 and h_2 as follows ($1 \leq k_1, k_2 \leq m$):

$$h(d) = x_{1k_1-1} \cdot x_{2k_2-1} \quad \text{if} \quad d \in [d_{1k_1-1}, d_{1k_1}) \text{ and } d \in [d_{2k_2-1}, d_{2k_2}) \tag{2}$$

Note that more than m AP-bounds for M can be computed from Equation 2. However, to be consistent with M_1 and M_2, we need to normalize function h so that it has only m AP-bounds. This can be done in a number of ways. One naive solution is to keep the first m bounds and throw the others away. With the help of hop functions, we get tighter AP-bounds and thus more MBRs could be pruned away using first-level pruning.

3 PNNT Query Processing

Before presenting our PNNT query processing algorithm, we first introduce the Global AP (GAP) function that is essential for pruning.

3.1 *GAP* Function

GAP function maintains the global AP information for the query point q. Let the distance to q be d. The definition of GAP function is as follows:

Definition 8. *GAP **function** $GAP(d)$ is the probability that no object exists inside the circle $C_{q,d}$.*

GAP is used to find and shrink the pruning circle as much as possible so that all MBRs outside of the circle can be pruned away. The radius R of the current pruning circle is maintained globally and decreases as more MBRs are seen during the query processing. GAP is updated whenever a new MBR is retrieved, whose absence probability contributes to GAP to make it more accurate. The algorithm *updateGAP* has the details. We use M to denote an MBR and $M.AP(q.threshold)$ to denote the AP-bound of M with bounding probability no greater than the query threshold. For each point on GAP function, we use d to denote the distance to query q and ap to denote $GAP(d)$, the absence probability of the circle $C_{q,d}$.

3.2 PNNT Query Processing Algorithm

Using the augmented R-tree index and the GAP function along with the pruning techniques we have introduced so far, we give the PNNT query processing algorithm here. The algorithm has two stages: Pruning stage and refining stage. In the pruning stage, the algorithm prunes away nodes in the augmented R-tree with the help of the GAP function. The goal of pruning is to dynamically update GAP as we see more MBRs so that we can shrink the pruning circle accordingly. The input of this stage is an augmented R-tree built upon all data items as well as the query point itself. The output of this stage is a list of data items that are NN candidates (obtained by applying pruning techniques) as well as non-candidates that overlap the final pruning circle. The reason such non-candidates are also returned is that they are useful in the refining stage for computing the exact NN probability of the candidates — when computing the NN probability for a data

Algorithm 1 Update GAP

Require: The current GAP, the query point (q), the newly-seen MBR (M)
Ensure: The updated GAP
  if $M.ap < q.threshold$ **then**
    set $currentPoint.d$ to be the distance between q and $M.AP(q.threshold)$
    if $currentPoint.d == d_{\max}(query, M)$ **then**
      $currentPoint.ap = M.ap$
    else
      $currentPoint.ap = q.threshold$
    end if
  end if // choosing a GAP point given M ends here
  if GAP is empty **then**
    add $currentPoint$ to GAP
  else
    $savedAP = currentPoint.ap$
    insert $currentPoint$ into GAP according to d, let the point before it be $prevPoint$
    if $currentPoint$ is not the first point of GAP **then**
      $currentPoint.ap = savedAp * prevPoint.ap$
    end if
    if there are points after $currentPoint$ in GAP **then**
      set their new ap to be the old ap times $savedAp$
    end if
  end if
  find the first point ($boundaryPoint$) in GAP with its $ap \leq q.threshold$
  set the pruning radius $R = boundaryPoint.d$
  discard all points in GAP with $d > R$

Algorithm 2 PNNT Query Processing

Require: The augmented R-tree ($tree$) for all data items, the query point ($query$)
Ensure: All data items with NN probability greater than $query.threshold$ ($results$)
  $prune(tree.root, query)$
  for each $node$ in non-discarded leaf-level nodes after pruning **do**
    for each data $item$ in $node$ **do**
      if $item$ is marked as 'c' (candidate) or 'k' (non-candidate to be kept) **then**
        add $item$ to $remains$ (non-discarded data items)
        if $item$ is marked as 'c' **then**
          add $item$ to $candidates$ (NN candidates)
        end if
      end if
    end for
  end for // pruning stage ends here
  for each $item$ in $candidates$ **do**
    $P_{nn} = computeNNProbability(item, remains, query)$
    if $P_{nn} > query.threshold$ **then**
      add $item$ to $results$
    end if
  end for
  return $results$ // refining stage ends here

item. all items that are not definitely further to the query must be taken into account [5]. The refining stage then decides whether a NN candidate is indeed a query result by checking whether its exact NN probability is greater than the threshold. The PNNT query processing algorithm is shown in Algorithm 2.

The details of the pruning algorithm (i.e. `prune`) are in Algorithm 3. The input is the query point and the node in the tree where the pruning starts. Note that we update the GAP function whenever we see a new MBR using algorithm `updateGAP` introduced in Section 3.1. `MarkMBRs` (Algorithm 4) marks all MBRs in the node as 'c', 'k' or 'd' according to the latest GAP function, where 'c' means NN candidates, 'k' means non-candidates that we need to keep for the refining stage and 'd' means others to be discarded.

Algorithm 3 Prune

Require: A node in *tree* (*node*) to start pruning, *query*
Ensure: All non-discarded nodes with marked MBRs
  for each MBR M in *node* **do**
    $updateGAP(M, query)$
  end for
  $markMBRs(node, query)$
  if *node* is a leaf **then**
    return
  end if
  $next = pickMBRtoExplore(node, query)$
  while *next* != NULL **do**
    $prune(next, query)$
    $markMBRs(node, query)$
    $next = pickMBRtoExplore(node, query)$
  end while

The nodes in the augmented R-tree are visited in a depth-first manner. `PickMBRtoExplore` picks an MBR in the node from all that are marked 'c'. The corresponding child of the node will then be explored. The criteria for picking is to choose the MBR that is furthest from the query point, in the hope that its children will be discarded soon.

4 Experimental Results

We performed our experiments on 1-dimensional data represented as intervals. Each interval is the uncertain region of the data and its pdf is represented using histograms. The total probability p over the interval is either in (0, 1] or in (0.5, 1], hence there is 1-p probability that the interval does not exist. The threshold τ of the PNNT query is at least 0.1, assuming that the user is not interested in small probabilities. All intervals are randomly generated within the range (0, 10000] and the size of the interval is in [1, 10]. For each experiment,

Algorithm 4 Mark MBRs

Require: A node in *tree* (*node*) to mark its MBRs, *query*
Ensure: All MBRs in *node* are marked
for each MBR M in *node* **do**
  if M is outside of the current pruning circle C_R centered at *query* with radius R **then**
    mark M as 'd' // first-level pruning
  else
    mark M as 'c'
    if $M.mp < query.threshold$ **then**
      mark M as 'k' // pre-pruning
    else
      search in GAP for the last point satisfying $GAP.d \leq d_{\min}(query, M)$
      if $M.mp * GAP.ap < query.threshold$ **then**
        mark M as 'k' //second-level pruning
      end if
    end if
  end if
end for

we average the statistics over 10 randomly generated query points in (0, 10000]. The default values/ranges for data size n, threshold γ and total probability p is 100000 (100K), 0.3 and (0, 1]. Our algorithm can handle arbitrary pdf represented by histograms. In the experiments, we generated data with either uniform pdf (default) or Gaussian pdf. In the following experiments, we always take default values of parameters unless otherwise specified. We ran our experiments (written in C++) on a PC with Intel T2500 2.00GHz CPU and 2.00GB main memory.

Our goal is to evaluate the performance of our PNNT query processing algorithm with different parameters presented below. We compared our algorithm with the naive algorithm that evaluates the exact NN probability for each data item and then returns all items with NN probabilities greater than γ. As the data size grows, the PNNT algorithm becomes more and more superior compared with the naive one. For 500 data items, the naive algorithm takes over 14.5 s while our algorithm takes no more than 3 ms, over 500 times faster. We also compared the performances of the three pruning techniques: Pre-pruning (`Prune0`), first-level pruning (`Prune1`) and second-level pruning (`Prune2`) (note that the actual algorithm combines all three together). For each point on the figures, we average its value over ten experiments with randomly generated query locations and draw the confidence interval associated with the value. The experimental results are as follows:

Effect of Data Size: We evaluated our algorithm by varying the data set size n from 10000 to 100000. We computed the pruning percentage of our algorithm by dividing the number of NN candidates (`candidateCount`) by n. Note that `candidateCount` is obtained after the pruning stage of the algorithm. We compared the pruning percentage when the total probability $p \in (0, 1]$ and

$p \in (0.5, 1]$. In the former case, p is just a random probability while in the latter case, p is above 0.5, indicating that the probability that the data item exists is greater than the probability that it does not exist. Fig. 4 shows the result: As data size grows, the pruning percentage also grows with a relatively sharp increase when the data size turns from 10000 to 20000. Over 99.7% data items are pruned away for both cases while a random $p \in (0, 1]$ generally has a higher pruning percentage than $p \in (0.5, 1]$. This might be due to the effect of pre-pruning that prunes away all data items with $p < 0.3$ ($\tau = 0.3$), which does not work at all for $p \in (0.5, 1]$. We also evaluated the time cost of our algorithm with regard to the pruning stage and the refining stage in Fig. 5. The time cost increases as data size grows. The pruning time cost is always bigger than the refining time due to the large data size (it takes longer to prune in the R-tree) as well as the high pruning percentage of our algorithm (there are less than 200 candidates to be evaluated even when the size reaches 100000). The total time cost (pruning and refining) is also shown in Fig. 5.

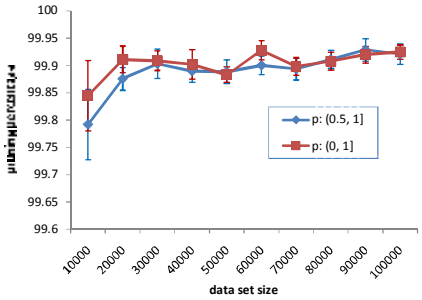


Fig. 4. Effect of Data Size on Pruning

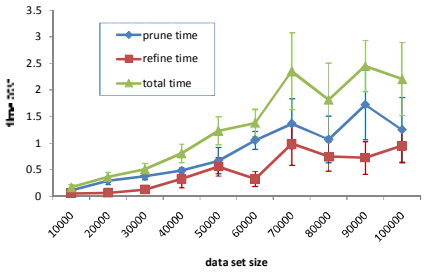


Fig. 5. Effect of Data Size on Time Cost

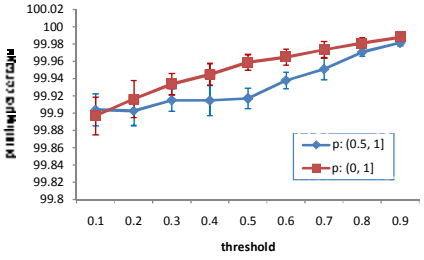


Fig. 6. Effect of Threshold on Pruning

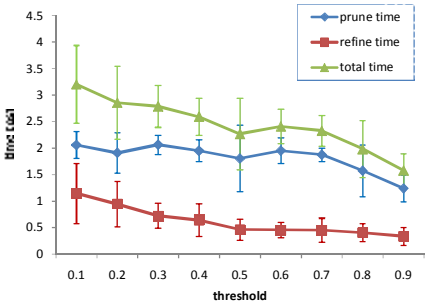


Fig. 7. Effect of Threshold on Time Cost

Effect of Threshold: We repeated the previous experiments with the data size fixed at 100000 and the threshold varying from 0.1 to 0.9 in Fig. 6 and Fig. 7.

Unlike the time cost experiment when the data size changes, the cost decreases as threshold increases. This is because we can prune much more when the threshold is big, as most of the data items are unlikely to have a NN probability above the big threshold. The pruning time is more than the refining time for the same reason as before. For the pruning percentage experiment, we observed similar results: The percentage increases as threshold becomes bigger and reaches as high as 99.98% when $\tau = 0.9$. The total probability $p \in (0, 1]$ still results in more pruning compared with $p \in (0.5, 1]$ like before. We further compared the three pruning techniques (`Prune0, Prune1, Prune2`) with the varying threshold in Fig. 8. The result shows that `Prune1` contributes the most of all three techniques with a pruning percentage around 99.8%, followed by `Prune0` and `Prune2`. As threshold increases, `Prune0` increases almost linearly when $\tau > 0.3$. `Prune1` stays relatively same with high pruning percentage while `prune2` increases slightly with fluctuations.

Data with Gaussian PDF: So far we have experimented with data that has a uniform pdf. We now show that our algorithm performs well for data with Gaussian pdf too. Fig. 9 shows the pruning percentage of the three pruning techniques over different thresholds. Compared with Fig. 8, we observe the similar results: `prune1` prunes most, followed by `prune0` and `prune2`. The pruning percentages of the three techniques are all above 94%.

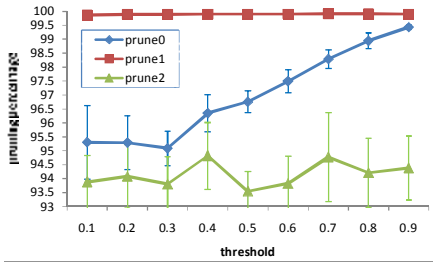


Fig. 8. Pruning Techniques

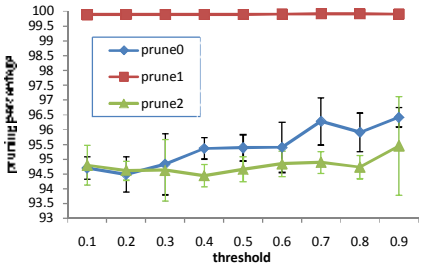


Fig. 9. Gaussian PDF

5 Related Work

Although the first probabilistic models were proposed as early as in the 1980's [1, 3], it is not until recently when many efforts have been made to design database systems that manage uncertainty [2, 4, 15]. Two major models for uncertainty exist: Tuple uncertainty models and attribute uncertainty models. For tuple uncertainty models [2, 4], each tuple is associated with a probability of its presence. For attribute uncertainty models, a tuple always exists, but there may be one or more uncertain attributes in the tuple with pdfs associated with them. Recently,

a database model is proposed for supporting pdf attributes that can handle both attribute and tuple uncertainty, which considers intra-tuple dependencies (captured by dependency sets) as well as inter-tuple dependencies (captured by history) [16].

In parallel to uncertainty modeling, much work has been done on indexing and querying uncertain data, including indexing for probabilistic threshold queries [7], indexing uncertain categorical data [14], processing range queries and PNN queries [10, 5], processing ranked queries [17, 12], etc. However, little work has been done on indexing and querying uncertain data with both attribute and tuple uncertainty which are unified in the model of [16]. In our paper, we proposed an efficient algorithm for processing PNNT queries. [5] gives an algorithm for PNN, but only prunes objects when their NN probability is 0. [8] proposed the concept of the "constrained PNN query" with a probability threshold and an error tolerance, which can be leveraged to prune objects away that cannot meet the threshold/tolerance requirement, thus saving many expensive computations of the exact NN probabilities. The authors followed the attribute uncertainty model and assumed that the probability of an object being in a closed region always adds up to 1, i.e., there is no tuple uncertainty – the object always exists. Our approach overcomes this limitation and can efficiently process PNNT queries when objects have both attribute and tuple uncertainty.

6 Conclusions

In this paper, we gave an efficient algorithm to process PNNT queries for uncertain data with missing probabilities, which is a problem that has not been addressed by any previous paper. Since data items that are close to the query point may not exist in this case, it is harder to prune away items that are further away (it still has a chance to be the NN). We designed an augmented R-tree index specifically for such queries. Each MBR is now associated with probability information such as MP, AP, and AP bounds to facilitate pruning. We further introduced a global data structure called GAP function to dynamically capture the decreasing of AP within the pruning circle. With this fine-grained AP information, we can apply our pruning techniques to prune away a large number of nodes in the R-tree. Our experiments proved the efficiency of the algorithm – over 90% data items can be pruned away, leaving only a small portion of data to be further examined as PNNT results.

References

1. D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowl. and Data Eng.*, 4(5):487–502, 1992.
2. Omar Benjelloun, Anish Das Sarma, Alon Halevy, and Jennifer Widom. Uldbs: databases with uncertainty and lineage. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 953–964, 2006.

3. Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, and Thomas Seidl. Fast nearest neighbor search in high-dimensional space. In *ICDE '98*, pages 209–218, 1998.
4. Jihad Boulos, Nilesh Dalvi, Bhushan Mandhani, Shobhit Mathur, Chris Re, and Dan Suciu. Mystiq: a system for finding more answers by using probabilities. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 891–893, 2005.
5. Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(9):1112–1127, 2004.
6. Reynold Cheng, Sarvjeet Singh, Sunil Prabhakar, Rahul Shah, Jeffrey Scott Vitter, and Yuni Xia. Efficient join processing over uncertain data. In *Proceedings of the 15th CIKM Conference*, pages 738–747, 2006.
7. Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proceedings of the 30th VLDB Conference*, 2004.
8. Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *Proceedings of the 24th ICDE Conference*, 2008.
9. Debabrata Dey and Sumit Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.
10. Dmitri V. Kalashnikov, Sunil Prabhakar, Susanne E. Hambrusch, and Walid G. Aref. Efficient evaluation of continuous range queries on moving objects. In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 731–740, 2002.
11. Jon M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *STOC '97*, pages 599–608, 1997.
12. Xiang Lian and Lei Chen. Probabilistic ranked queries in uncertain databases. In *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, pages 511–522, 2008.
13. Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD*, 1995.
14. Sarvjeet Singh, Chris Mayfield, Sunil Prabhakar, Rahul Shah, and Susanne Hambrusch. Indexing uncertain categorical data. In *Proceedings of the 23rd ICDE Conference*, 2007.
15. Sarvjeet Singh, Rahul Shah, Sunil Prabhakar, and Chris Mayfield. Unified model for uncertainty management in databases. volume 21, pages 560– 571, 2003.
16. Sarvjeet Singh, Rahul Shah, Sunil Prabhakar, and Chris Mayfield. Database support for pdf attributes. In *Proceedings of the 24th ICDE Conference*, 2008.
17. Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. Urank: formulation and efficient evaluation of top-k queries in uncertain databases. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1082–1084, 2007.

Measuring and Constraining Data Quality with Analytic Workflows

Laure Berti-Équille [*]
Université de Rennes 1
Campus Universitaire de Beaulieu
35042 Rennes cedex, France
berti@irisa.fr

ABSTRACT

One challenging aspects of data quality modeling and management is to provide flexible, declarative and appropriate ways to express requirements on the quality of data. The paper presents a framework for specifying and checking constraints on data quality in RDBMS. The evaluation of quality of data (QoD) is based on the declaration of data quality metrics that are computed and combined into so-called *QoD analytic workflows*. These workflows are designed as a composition of statistical methods and data mining techniques used to detect patterns of anomalies in the data sets. As metadata they are used to characterize various quantifiable dimensions of data quality (e.g., completeness, freshness, consistency, accuracy). The paper proposes a query language extension for constraining data quality when querying both data and its associated QoD metadata. Probabilistic approximate constraints are checked to determine if the quality of data is (or not) acceptable to build quality-constrained query results.

1. INTRODUCTION

With the ever-growing data glut problem, our capabilities for collecting and storing data have far outpaced our abilities to analyze, summarize available data, and more critically, to evaluate and systematically check the quality of this data (QoD). While database technology has provided us with the basic tools for the efficient storage and lookup for large data sets, one of the current issues is how to measure, analyze and enforce data quality in databases. Data quality is known as a "multidimensional, complex and morphing concept" [10]. Maintaining a high level of data quality in a database is challenging and cannot be limited to one-shot approaches addressing simpler and more abstract versions of the wide range of data quality problems. Technically, data quality management mainly refers to the detection and elimination of various data quality problems, such as:

- *Duplicate and redundant data.* A wide range of techniques have been proposed for record linkage [23] and entity resolution since the "merge/purge problem" identified by Hernández and Stolfo [12],
- *Imperfect data.* Inconsistency, imprecision, and uncertainty are some of the problems associated with data imperfection [9]. A significant amount of work has been proposed in the areas of integrity constraint checking, imprecise data management, and uncertainty in DBMSs, e.g., [2, 4, 9],
- *Missing values and incomplete database.* The problem of handling incomplete information has also been addressed in information and database systems [17],
- *Stale data.* Various refreshment techniques and synchronization policies have been proposed for ensuring data freshness depending on the type of system architecture [6, 20]: e.g., data warehousing systems check the recentness of materialized views; caching systems estimate the time-to-live of cached data before expiration and tune the caching policy for ensuring data currency.

QoD dimensions are numerous with various definitions, interpretations and measurement methods depending on the considered application domain. One of our goals is to cope with this diversity in proposing a flexible way to express and measure the dimensions of data quality and to check constraints on these dimensions. Among other challenging research directions that have been recently identified in Data Quality Research [5] (e.g., pattern design for quality-aware IS engineering [1], new perspectives of methodological approaches for data quality management [3] or benchmarks for comparing objectively the research contributions to specific data quality problems [22]), QoD metadata modeling and management and QoD-aware query languages are indeed of particular interest for the community. There is currently no model that captures in a precise and easy way the semantics of all static and dynamic dimensions of data quality, and allows carrying out relevant automatic checking based on QoD metadata management. The existing propositions focus on one or two *hard-coded* QoD dimensions (often considered separately) [14, 11, 13, 20]. The approach which consists in defining *default* QoD dimensions and in basing

[*]Currently visiting researcher at AT&T Labs Research, NJ, USA supported by a Marie Curie International Fellowship within the 6th European Community Framework Programme (FP6-MOIF-CT-2006-041000).

VLDB '08, August 24-30, 2008, Auckland, New Zealand

the analysis on the Cartesian product or on the *ad-hoc* composition of some data quality dimensions do not make it possible to model, in a faithful way, the interdependencies between these QoD dimensions [1]. It is thus necessary to develop metadata models and declarative data manipulation languages which make it possible to represent combinations and interdependencies of user-defined QoD dimensions with relevant measurement techniques and constraints. As a first step in this direction, the paper presents a framework and a QoD-constrained language for integrating "natively" the specification, evaluation and checking of data quality in the data management system [5]. The emphasis of the paper is put on:

- **QoD Evaluation**. We define and implement *QoD analytic workflows* for computing of a set of statistical and data mining functions that evaluate various dimensions of data quality. The results of these functions are stored in a metadata repository as QoD measures characterizing data distribution properties, data quality problems, anomaly patterns, and any useful information reflecting QoD evaluation for different granularity levels of a RDBMS (*i.e.,* for the values, tuples, attribute domains, tables or the database),
- **QoD-aware data management**. We present a query language extension that allows the declaration and assignment of *contracts* on the quality of data as sets of constraints on the QoD measures resulting from the analytic workflows.

The paper is organized as follows: Section 2 presents the main steps of our approach based on the design of analytic workflows for QoD evaluation. Section 3 presents the syntax of the language extension we propose for measuring and checking constraints on data quality. Section 4 presents related work on declarative languages dedicated to data quality control and management. Section 5 concludes the paper and presents our research perspectives.

2. A FRAMEWORK FOR EVALUATING AND CHECKING QOD

2.1 Overview

Our approach can be summarized through the following steps:

1. **Definition and computation of analytical functions for QoD evaluation**. First, the design of analytic workflows consists of the definition (or eventual reuse) of functions that measure objectively various dimensions of the quality of data (e.g., freshness, consistency, completeness, accuracy); each dimension may characterize an aspect of the quality of a database object instance (*i.e.*, value, tuple, attribute domain, table or database). The computed measures are stored and managed as QoD metadata in a repository. One QoD dimension of a DB object instance can be characterized by many QoD measures in the metadata repository.
2. **Definition of probabilistic and approximate constraints on QoD measures**. To establish a QoD diagnostic, measured QoD values have to be compared to expected QoD values with a certain degree of tolerance. To do so, probabilistic and approximate constraints are specified and checked to determine if the quality is acceptable or not for each QoD dimension. The computed probabilities are stored in the metadata repository and refreshed by the system.
3. **Quality-constrained query declaration and processing**. Probabilities assigned to the QoD dimensions are then used in the query processing to build quality-aware query results (or QoD diagnostics) in conformance with the QoD constraints predefined in the analytic workflows.

From the system-centric perspective, our approach has been translated into three main components that pragmatically integrate data quality awareness in the system: *i)* an extensible library of functions and statistical methods for measuring various aspects of QoD, *ii)* a quality metadata manager that stores, indexes, searches, and refreshes QoD measures and related descriptive metadata, and *iii)* an extended query engine that allows declaration and manipulation of data with probabilistic approximate constraints on QoD metadata. The paper mainly focuses on the components *i)* and *iii)* for declaring and manipulating constraints on QoD.

2.2 Designing QoD Analytic Workflows

In order to analyze and evaluate the quality of data, a relevant sequence of tasks are specified and planned depending on the underlying goal of the QoD evaluation. For this purpose, we define *QoD analytic workflows* as sets of parameters interacting with each other. These include: a goal (G), a set of tasks (T), a set of resources (R), a set of function allocations (A), and a set of limitations (L). A QoD analytic workflow, W, is then a function of all the sets interacting with each other, as: $W = \{G, T, R, A, L\}$. The goal G of an analytic workflow is defined for a DB object instance resource in order to characterize it and to analyze a variety of data quality issues for a specific purpose. A goal has three components: *i)* the set of resources (in our case, data instance objects such as a set of rows or a set of values depending on the considered granularity), *ii)* the data quality issue (e.g., data consistency, freshness or completeness), and *iii)* the purpose (e.g., detect outliers or correct anomalies). The set of tasks (T) are the building blocks of the analytic workflow. Tasks can be broken down into smaller tasks through task refinement. This activity continues until a satisfied level of abstraction has been achieved for that particular QoD analytic workflow being modeled. The set of resources (R) include the set of inputs of the analytic workflow, such as the set of DB object instances to analyze (e.g., attribute domain, tuple, value) and the set of output QoD metadata including QoD measures and descriptive metadata (*i.e.,* detailed description of the settings and parameters of the functions). The set of function allocations (A) define the set of analytical functions to be applied for QoD evaluation. A set of limitations (L) may be specified and associated to a task or to a function allocation stating for example that the function can only be applied to categorical data sets.

Table 1 gives a classification of the analytical functions that are used for QoD evaluation; the mention from level I to level IV indicates the increasing range of complexity of the methods. It also provides examples of the functions used

Level	Category	Description	Examples		
			QoD	Functions	DB object
I	QoD Profiling Functions	Simple counts computed from the database dictionary, look-up tables, control log or trace files, usually with single-pass algorithms. These metadata can be used to characterize some aspects of database completeness and freshness depending on the level of granularity of the database object instances (*i.e.*, value, record, column, table or database in the relational context)	CP	`nullValues%` returns a percentage that represents the quantity of null values.	D,T,R,A
			F	`updateFreqStdDev` returns a decimal in [0,1] that represents how frequently the DB object instance has been updated since its creation time compared to the average frequency of the objects of the same granularity.	D,T,R,A
II	QoD Constraint-Based Functions	Rules or rules of application-specific business logic, consistency constraints or inferred rules from statistical techniques that characterize the most plausible relationships between data instances or that compute the deviations from the rules. These constraints are verified at runtime. Constraint violations indicate errors or dubious data.	CT	`SyntacticCorrectness` returns a decimal number in [0,1] that represents the percentage of format discordances, syntactical errors and misspellings.	D,T,A,R,V
III	QoD Synopses Functions	Statistical summaries, aggregates or parametric estimations computed as approximate answers with deterministic error bounds or probabilistic guarantees that the approximate answer is the actual one. Basic synopses are samples, equi-depth histograms, quantile computation. Advanced synopses are computed from sketch-based computation techniques (e.g., V-optimal histograms, wavelets). They are useful to quickly reveal unlikely values that are artifacts or inconsistent patterns from samples of very large data sets before going into deeper and more expensive analysis.	AC	`outlierProb` returns a decimal number in [0,1] based on IQR that represents the probability of being an outlier in the attribute domain.	V
IV	QoD Exploratory Mining Functions	Data mining results obtained from techniques such as clustering, association rule discovery, and decision trees. These techniques have the same goal as the previous methods, in the sense that they can be used to detect data glitches (e.g., duplicates, anomaly patterns, and dubious data), but their computation and maintenance costs are much higher.	U	`DupDetectionProb` returns a decimal number in [0,1] that represents the probability of the DB object instance to be a duplicate after clustering and association rule mining on a combination of attributes whose values are identical or similar.	R

Table 1: Categories of Functions for QoD Evaluation

for computing measures related to the following data quality dimensions: completeness (CP), consistency (CT), accuracy (AC), freshness (F) and uniqueness (U) (*i.e.,* absence of duplicates). The computed measures are respectively associated to the instances with the level of granularity the measure is computed from, namely the value (V), record (R), attribute (A), table (T) and database (D) (see the last column of Table 1). The granularity levels correspond to the main structural elements of the relational DB; metadata are thus associated to each DB object instance depending on its granularity level.

The reader is invited to read the chapter 2 of [5] to have a more detailed description of the functions that are used in the QoD analytic workflows.

The set of resource allocations (A) defines the relationship of the tasks and the relationship of the resources, as well as the task/resource allocations. These relationships are given at the time the resources and tasks are defined. The set of limitations (L) defines any limitations or restrictions imposed on the tasks and resources. Such restrictions may include scheduling restrictions (with precedence constraints in task planning), resource restrictions (e.g., resources A and B are mutually exclusive), resource allocation restrictions, (e.g., a function f cannot be applied for task t), and so on. Once sets of resources and their relations are defined in the QoD analytic workflow model, the data and functions become inputs to the tasks and QoD metadata become output. Output from one analytical function could serve as input to other functions; the same resource can serve either as input or output, or both depending on which task it applies to.

Figure 1 presents the example of a QoD analytic workflow designed for evaluating the quality of data of the database named CRM_DB composed of two tables: PRODUCT and CUSTOMER. This includes several tasks of evaluation on three QoD dimensions, namely freshness, accuracy, and completeness at different granularity levels (e.g., cell, row, column, table). Data object instances are the inputs of the tasks. Each task may be composed of several subtasks with allocated functions for the computation of the QoD measures. Summary statistics for numerical values are computed, out-of-range data values may also be detected with univariate statistics and percentiles (IQR) e.g., '`sas_func_iqr_outlierProb.sas`' of the subtask `outlierProb` composing the task `ACCURACY` evaluates the accuracy at the `CELL` granularity level of CRM_DB. The execution of each allocated function generates QoD measures that are stored in the metadata repository as arrays of QoD values represented in Figure 1 as **{Q}**. The detailed description of each function is stored as descriptive metadata in the repository using

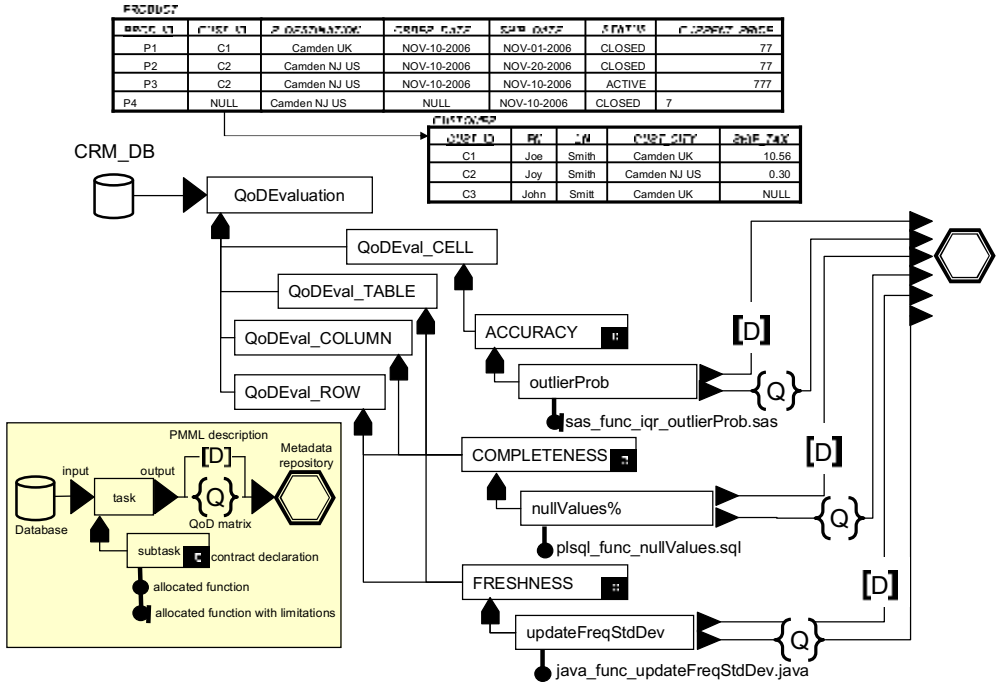


Figure 1: Example of QoD Analytic Workflow for CRM_DB

PMML[1] represented as [**D**] in Figure 1. PMML is used to describe in details the parameters, thresholds or initial conditions that are necessary for executing the specified analytical functions. For traceability reasons, the PMML descriptions of the functions are stored in the metadata repository mainly to allow the users to keep the complete and detailed history of the functions used in the workflows and also to be able to eventually change the initial parameters or thresholds for computing new QoD measures.

Analytic workflows specify and combine multiple analytical functions as the ones illustrated in the example. Functions may be implemented in different ways (stored procedure, SAS script, C or Java program) depending on the set of limitations specified in the workflow. Of course, many other analytical functions may be added to the library. The panel of analytical functions used for computing QoD measures and generating QoD metadata gives relevant indications for characterizing potential data quality problems and understanding anomaly patterns. Nevertheless, the overall process is a 2-step based approach: in the first off-line step, the analytic functions declared in the contract types of the workflow compute QoD measures from the database object instances; these measures are stored in the metadata repository and in the second step, constraints on data quality are declared in the contract instances or in the QWITH queries and they compare the pre-computed measures from the metadata repository to expected thresholds in order to build the query result.

[1]PMML - Predictive Model Markup Language, Version 3.1: http://www.dmg.org/

3. DECLARATION OF CONSTRAINTS ON DATA QUALITY

In our approach, constraints on data quality are grouped and expressed by means of *quality contract types* and *quality contracts instances*:

1. A **quality contract type** defines a set of quality dimensions, measures and functions associated to a particular database object instance. It corresponds to the computation tasks of a particular QoD analytic workflow.

2. A **quality contract instance** is a set of one-sided range constraints defined on the QoD dimensions declared in the contract type.

3.1 Declaration of QoD Contract Types

The syntax for creating a quality contract type is given in Figure 2. A contract type is named (`ct_name`) and assigned to a given schema of the database. Each contract type is composed of a list of named measurable dimensions, the datatype of the output result, the granularity level which the measure is associated to (e.g., `ON CELL, ROW, COLUMN, TABLE, DATABASE`) and the identifier or name of the function that computes the measure following the `BY FUNCTION` statement. QoD measures are stored in the metadata repository and they are assigned either to: *i)* a global granularity level, *i.e.* `ON CELL, ROW, COLUMN, TABLE`, or `DATABASE`, or to: *ii)* a specific DB object instance, *i.e.*, an existing table, column, record, or cell. For each row in the database, the ROWID pseudo column returns a row's address.

The analytical function may be a PL/SQL[2] procedure or the call specification of a Java, C or SAS program. The creation of a

[2]Although our implementation was initially made to access Oracle databases with providing a set of appropriate

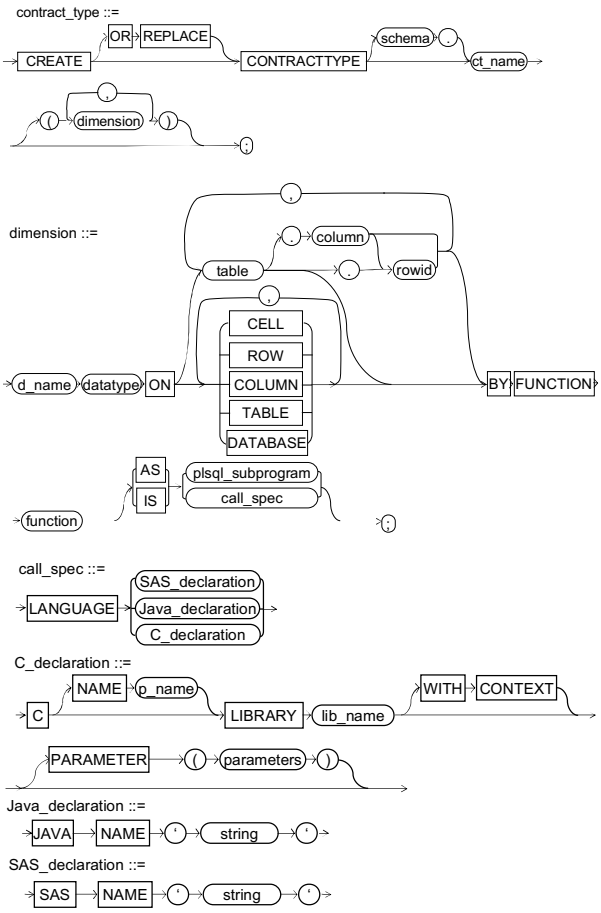


Figure 2: Syntax of Quality Contract Type Creation

task in the QoD analysis workflow can be associated to the creation of a contract type on a database object instance. This leads to the execution of the declared analytical functions, the computation and storage of the measured QoD values and the associated PMML descriptions of the functions in the metadata repository. As an example, Table 2 gives the syntax for creating the quality contract types resulting from the tasks of the illustrated QoD analysis workflow.

For instance, in `FRESHNESS` contract type, `updateFreqStdDev` is a local procedural type in JAVA function names: '`java_func_updateFreqStdDev.java`' that indicates how frequent the rows and tables have been updated since their creation. In `ACCURACY` contract type, `outlierProb` is a float returned by a SAS function named '`sas_func_iqr_prob.sas`' that returns the probability of being outlier for each value of the database. In `COMPLETENESS` contract type, the PL/SQL procedure `plsql_func_nullValues` returns a percentage of missing values for each row, column, and table of CRM_DB.

3.2 Declaration of QoD Contract Instances

The syntax of creation of a quality contract is given in Figure 3. Each contract declaration refers to an existing contract type (`ct_name`). It defines the constraints on each contract type

PL/SQL stored procedures for characterizing various data quality dimensions, an interesting extension would be to provide the ability to the user to call or declare any stored procedure in the proprietary language of the database management system he/she is connected to.

```
CREATE CONTRACTTYPE FRESHNESS(
 updateFreqStdDev FLOAT ON ROW, TABLE
  BY FUNCTION java_func_updateFreqStdDev
  IS LANGUAGE JAVA
  NAME 'java_func_updateFreqStdDev.java');
CREATE CONTRACTTYPE ACCURACY(
 outlierProb FLOAT ON CELL
  BY FUNCTION sas_func_iqr_prob
  IS LANGUAGE SAS
  NAME 'sas_func_iqr_prob.sas');
CREATE CONTRACTTYPE COMPLETENESS(
 nullValues% FLOAT
   ON ROW, COLUMN, TABLE
   BY FUNCTION IS plsql_func_nullValues);
```

Table 2: Example of Quality Contract Type Declaration

dimension with simple or composed expressions using basic binary operators. Consider again the CRM_DB database and the QoD analysis workflow given in Figure 2. Three quality contract types have been defined for characterizing freshness, accuracy, and completeness on CRM_DB.

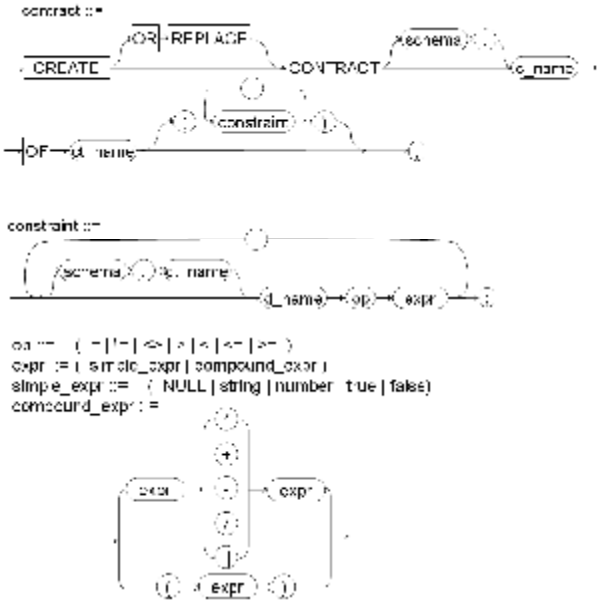


Figure 3: Syntax of Quality Contract

Table 3 gives examples of quality contract instances, named `fresh`, `accurate`, and `complete` whose definition is based on their respective contract types: `FRESHNESS`, `ACCURACY`, and `COMPLETENESS` given previously. Contract instances describe the one-sided range constraints to be checked in conformance with the expected values on each dimension declared in the corresponding quality contract type.

```
CREATE CONTRACT fresh OF FRESHNESS(
   updateFreqStdDev > (.50,.50));
CREATE CONTRACT accurate OF ACCURACY(
   outlierProb < .06;
CREATE CONTRACT complete OF COMPLETENESS(
   nullValues% <= (.20,.20,.35);
```

Table 3: Example of Contract Declaration

The declaration of contract types and instances is a part of the QoD analytic workflow specification. Our objective is to incorporate and use a set of analytical functions for computing QoD measures that can be easily extended by end-user defined functions. The call and execution of functions is triggered immediately after the validation of the contract type declaration in the workflow design.

3.3 Checking Constraints on QoD Measures

Once the contracts type and instances have been declared and enforced in the workflow execution, constraints on QoD measures are checked at the specified granularity levels or for the particular database object instances. The constraints are based on the comparison between measured and expected values for each QoD dimension. The degree to which these constraints are satisfied is computed as an aggregated value called *QoD acceptability* combining the checking result of the constraints on every QoD dimension. This value represents the probability that the QoD dimension is acceptable considering that the values of its associated QoD measures are in conformance with expected bounds given in the constraints of the contract instance. More formally, given a particular QoD dimension Q_i, the QoD acceptability of a DB object instance indicates the likelihood δ_i that the QoD dimension measured for the database object instance o is acceptable considering the constraints c_{ij} on its associated QoD measures m_{ij} (using an index with respect to an interval of expected values M_{ij} with a tolerance τ_{ij}. Acceptability of QoD dimension Q_i on the DB object instance o, noted $Acc_i(o)$ is given below:

$$Acc_i(o) = Pr(Q_i(o) | \bigwedge_j c_{ij} : m_{ij} \in [M_{ij} - \tau_{ij}]) = \delta_i. \quad (1)$$

Because all the facets (and corresponding measures) of a QoD dimension may not be equally considered, δ_i is defined as the weighted sum of all distances between measured and expected values that characterize the QoD dimension Q_i, as:

$\delta_i = \sum_j w_{ij}.d_{ij}$ with $\sum_j w_{ij} = 1$ and

$$d_{ij} = \begin{cases} 0 & \text{if } c_{ij} \text{ is satisfied with } \tau_{ij} = 0 \\ Normdist(m_{ij}, M_{ij}) & \text{if } c_{ij} \text{ is satisfied with } \tau_{ij} > 0 \\ 1 & \text{if } c_{ij} \text{ is not satisfied with } \tau_{ij} > 0 \end{cases}$$

d_{ij} is null when the measured QoD value m_{ij} satisfies the constraint c_{ij} with respect to the expected interval M_{ij} without τ_{ij}. If the constraint is not satisfied even within the interval of the tolerance noted $[M_{ij} \pm \tau_{ij}]$, then d_{ij} equals 1; otherwise d_{ij} is computed as the normalized distance between the measured QoD value and the expected value for the object instance o. Suppose that a row has the measure for `updateFreqStdDev` equals to .42 which does not satisfy exactly the declared constraint ($.42 < .50$), the possibility to soften the constraint may be interesting in order to provide results to the user that do not exactly meet their requirements in terms of quality but still are close.

Suppose the quality contracts given in Table 3 are applied to CRM_DB. For each CRM_DB object instance, the constraints declared in the quality contracts are checked. For each granularity level, Table 4 gives the computed probabilities for the DB object instances of the PRODUCT table of CRM_DB. Again, null probability means acceptable QoD dimension; 1 means unacceptable. We suppose that the probabilities of the other CRM_DB object instances are null.

3.4 Constraining Data Quality in the query

We have designed a query language extension named XQuaL for manipulating and checking constraints on the quality of data in RDBMS. The syntax of a quality-extended query is given in Figure 4. Once declared, one (or several) contract(s) or constraints may be used in the QWITH part of the XQuaL queries.

`'sfw_query'` represents the classical SELECT-FROM-WHERE statement of the query. In the QWITH statement, declared contract instances are identified and the constraints defined in the declared contracts are checked on ROW or CELL granularity levels specified after the statement ON. In the QWITH declaration, two alternatives are possible for checking constraints:

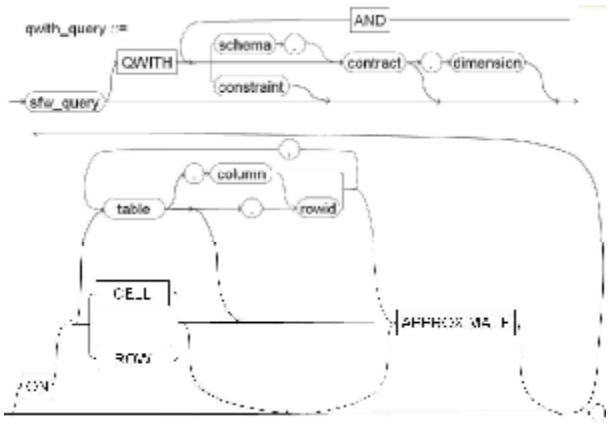


Figure 4: Syntax of QWITH queries

- *Exact checking (default).* Only DB object instances that exactly satisfy (with $\tau = 0$) the constraints declared in the contracts are used in the query processing for building the query result,
- *Approximate checking.* DB object rows and cells that satisfy approximately (with non null τ) the constraints defined in the contracts of the QWITH query will be considered for elaborating the query result.

Consider the query that retrieves all the products whose PRICE is greater than $10 in CRM_DB. QoD metadata of PRODUCT table involved in the query have been defined by means of the quality contract types and instances previously declared in Tables 2 and 3. A *quality-blind* query would return three rows: $P1$, $P2$ and $P3$. Different quality aware queries are given in Tables 5 and 6 based on the declared quality contracts that check data freshness, accuracy, and completeness at the ROW and CELL granularity levels. Results are presented in both EXACT and APPROXIMATE modes along with the acceptability values of the result for each QoD dimension considered in the QWITH part of the query. These queries lead to different results depending on the constraints required on the chosen QoD dimensions.

In the EXACT mode, only DB object rows and cells that satisfy exactly (with $\tau = 0$) all the constraints defined in the contracts invoked in the QWITH query will be considered for elaborating the query result. For Q1 query, the contract `fresh` is defined on two granularity levels (ROW, TABLE) as given in Table 2. The constraints are checked on data freshness for each cell and each row involved in the SFW query processing. Probabilities resulting from the checking are given in the fourth column of the tables. In the EXACT constraint checking mode, only the rows and cells involved in the query with null probability will be considered for building the query result. Consequently, $P1$ row will be excluded ($Pr_{fresh}(ROW('P1')) = .42$ in Table 4) whereas in the APPROXIMATE mode, $P1$ row will be included in the result ($.42 + .15 > .50$).

For Q2, the query result will not include product $P3$ in the EXACT mode and also in the APPROXIMATE mode as well because its PRICE value ('7777') has a high probability of being inaccurate with respect to the contract `accurate` applied to CELL granularity level ($Pr_{accurate}(CELL('7777')) = .35$ in Table 4). The result presentation includes the probabilities of every cell returned in the result.

For Q3 constraining freshness and completeness on rows and accuracy on cells, only $P2$ satisfies the constraints and the query result will exclude $P1$ and $P3$ in the EXACT mode but it will include $P1$ in the APPROXIMATE mode.

Consider a query that retrieves the set of product identifiers and the city of their purchaser, PROD_ID and CUST_CITY joining PRODUCT and CUSTOMER tables on CUST_ID. A

	CELL	ROW		COLUMN		TABLE
Acceptability	**7777**	*P*1	*P*4	**CUST_ID**	**ORDER_DATE**	**PRODUCT**
P_{fresh}	-	.42	0	0	-	.56
$P_{accurate}$	.35	-	-	.06	-	-
$P_{complete}$	-	0	.286	.25	.25	.071

Table 4: Acceptability Values per QoD Dimension in PRODUCT tables of CRM_DB

Q#	Query	Results			
		Exact		Approximate ($\cdot$ = .15)	
Q1	SELECT PROD_ID, PRICE FROM PRODUCT WHERE PRICE > 10 QWITH fresh ON ROW;	P2,77 P3,7777	$P_{fresh}(ROW(\text{P2})) = 0$ $P_{fresh}(ROW(\text{P3})) = 0$	P1,77 P2,77 P3,7777	$P_{fresh}(ROW(\text{P1})) = .42$ $P_{fresh}(ROW(\text{P2})) = 0$ $P_{fresh}(ROW(\text{P3})) = 0$
Q2	SELECT PROD_ID, PRICE FROM PRODUCT WHERE PRICE > 10 QWITH accurate ON CELL;	P1,77 P2,77:	$P_{accurate}(CELL(\text{P1})) = 0$ $P_{accurate}(CELL(77)) = 0$ $P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$	P1,77 P2,77:	$P_{accurate}(CELL(\text{P1})) = 0$ $P_{accurate}(CELL(77)) = 0$ $P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$
Q3	SELECT PROD_ID, PRICE FROM PRODUCT WHERE PRICE > 10 QWITH fresh ON ROW AND accurate ON CELL; AND complete ON ROW;	P2,77	$P_{fresh}(ROW(\text{P2}))) = 0$ $P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$ $P_{complete}(ROW(\text{P2})) = 0$	P1,77 P2,77	$P_{fresh}(ROW(\text{P1})) = .42$ $P_{accurate}(CELL(\text{P1})) = 0$ $P_{accurate}(CELL(77)) = 0$ $P_{complete}(ROW(\text{P1})) = 0$ $P_{fresh}(ROW(\text{P2}))) = 0$ $P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$ $P_{complete}(ROW(\text{P2})) = 0$

Table 5: Examples of simple QWITH Queries on PRODUCT table

quality-blind query would return $P1$, $P2$, and $P3$ PROD_IDs. Different quality-aware queries may be formulated with respect to the previous quality contracts on freshness, accuracy, and completeness.

Suppose that all the acceptability values of CUSTOMER table are null except for cell $C2$ and its associated row such as $P_{fresh}(CELL('C2')) = .6$, $P_{accurate}(CELL('C2')) = .2$ and $P_{complete}(ROW('C2')) = .5$.

For Q4 in the EXACT mode, the join of PRODUCT and CUSTOMER tables only consider the joining cells that have a null probability of being inaccurate with respect to the contract `accurate`, then $P2 \quad C2$ and $P3 \quad C2$ joins will be excluded from the join result because the probability of $C2$ cell from CUSTOMER table is $Pr_{accurate}(ROW('C2') = .2$ with respect to contract `accurate`. But the in the APPROXIMATE mode, this cell will be considered for joining the tables and leads to a result including $P2 \quad C2$ and $P3 \quad C2$ rows.

For Q5, the join of PRODUCT and CUSTOMER tables only consider the rows of both tables having null probability with respect to the contracts `fresh` and `complete`. $P1$ will then be rejected in the EXACT mode but included in the APPROXIMATE mode ($Pr_{fresh}(ROW('P1') = .42 < .50$ in Table 4).

4. RELATED WORK

In Data Quality Research, several contributions have been proposed for expressing in a simply and declarative way constraints on data quality and extending the query languages. Different approaches have used data quality indicators and metadata for selecting the best source [15] or the best query plans to execute it in a distributed environment. As the first prototype, Q-Data [21] checks if the existing data is correct and ensures data validation and cleanup by using a logical database language (LDL++). More recently, Guo et al. [11] have proposed a model for expressing currency and consistency constraints (C&C) in the queries on replicated and cached data by the means of a new clause on data currency that extends SQL. DQ^2L (*Data Quality Query Language*) [14] was designed to query relational data supporting a data quality aware query processing framework. In the context of quality-driven query processing, Braumandl et al. [8] propose to take into account data quality estimates when evaluating the user's query and deciding the best manner of carrying out the query (which sources to reach, which server to use, etc). TriQL[3], the query language of the Trio project [16] is an extension of SQL including some built in predicates for evaluating confidences, accuracy and lineage of queried data and extending traditional data management.

All these approaches have proposed an extension of SQL query language in order to include in different ways constraints on specific and fixed dimensions of the quality of data used by the query answering. They may differ from the one used here but they are neither flexible nor modular from the user perspective in the way that QoD is evaluated by hard-coded functions. In our approach, an extensible library of functions can be used to define and compute QoD measures. The composition of functions is required through the design of analysis workflows that can be set up by the user to automatically check and enforce various aspects of data quality as it is clearly important for his/her specific needs and goals. In terms of development, XQuaL prototype is currently being implemented on Eclipse Platform 3.3.2 (JRE1.5.0_12) using SQLExplorer Plug-in 3.5.0 and based on Kepler[4][7] that is built upon Ptolemy II framework[5] developed at the University of California, Berkeley. Kepler is a scientific workflow management system that allows specifying and executing data-intensive analysis. A workflow in Kepler can be graphically designed by connecting together tasks, where each task may take input data from previous tasks, parameter settings, and data coming from external data sources. The contribution of the XQuaL project in terms of development concerns: *i)* the implementation of the extended QWITH query engine, *ii)* the development of the library of functions for QoD evaluation, *iii)* a QoD contract editor invoking C, SAS or PL/SQL functions extending the design of Kepler workflows for QoD analysis. *iv)* the coupling of the QWITH query engine with the workflows and the specification of contracts types and instances, and *v)* the development of a QoD metadata repository for storing QoD measures.

[3]TriQL : http://infolab.stanford.edu/widom/triql.html
[4]Kepler project: http://www.kepler-project.org/
[5]Ptolemy II: http://ptolemy.eecs.berkeley.edu/ptolemyII/

Query#	Query	Results			
		Exact		Approximate ($\epsilon = .15$)	
Q4	SELECT PROD_ID, CUST_CITY FROM PRODUCT P, CUSTOMER C WHERE P.CUST_ID=C.CUST_ID QWITH accurate ON CELL;	P1,Camden UK	$P_{accurate}(CELL(\text{P1})) = 0$ $P_{accurate}(CELL(\text{Camden UK})) = 0$	P1,Camden UK	$P_{accurate}(ROW(\text{P1})) = 0$ $P_{accurate}(ROW(\text{Camden UK})) = 0$
				P2,Camden NJ US	$P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(\text{Camden NJ US})) = 0$
				P3,Camden NJ US	$P_{accurate}(CELL(\text{P3})) = 0$ $P_{accurate}(CELL(\text{Camden NJ US})) = 0$
Q5	SELECT PROD_ID, CUST_CITY FROM PRODUCT P, CUSTOMER C WHERE P.CUST_ID=C.CUST_ID QWITH fresh ON ROW AND complete ON ROW;	P2,Camden NJ US	$P_{fresh}(ROW(\text{P2})) = 0$ $P_{complete}(ROW(\text{P2})) = 0$	P1,Camden UK	$P_{fresh}(ROW(\text{'P1'})) = .42$ $P_{complete}(ROW(\text{P1})) = 0$
		P3,Camden NJ US	$P_{fresh}(ROW(\text{P3})) = 0$ $P_{complete}(ROW(\text{P3})) = 0$	P2,Camden NJ US	$P_{fresh}(ROW(\text{P2})) = 0$ $P_{complete}(ROW(\text{P2})) = 0$
				P3,Camden NJ US	$P_{fresh}(ROW(\text{P3})) = 0$ $P_{complete}(ROW(\text{P3})) = 0$

Table 6: Examples of QWITH JOIN Queries

5. CONCLUSION AND PERSPECTIVES

Since real-life applications need tools to evaluate with certainty the quality of data in a large database, we propose an analytical and probabilistic approach that allows the evaluation of the quality of database object instances and takes into account this evaluation for elaborating and presenting alternative query results depending on constraints on data quality. Since it is also important to accommodate *ad-hoc* queries with taking into account possible data quality requirements, which of course, *a priori* are unknown, our objective is to evaluate data quality in the context of known uses (e.g., at the query time). In this paper, we address this by proposing a complete approach based on the use of various functions, statistics, and data mining techniques that are combined into analytic workflows devoted to QoD evaluation for quality-aware query processing and QoD diagnostics. Analytical functions generate measures that are modeled by ontologies and defined dimensions of QoD. These measures are stored as metadata in a repository. They are checked with respect to user-defined constraints that are associated to the database object instances and refined by means of rewrites. Several QoD measures may be associated to one QoD dimension. A scoring function is used to compute the probability that for a given DB object instance, a QoD dimension is acceptable with respect to a quality contract that defines the set of constraints on its associated QoD measures.

QoD evaluation can be based on the results of typical data-centric analysis. They require a sequence of activities and components for data retrieval computation, and presentation, assembled together into a single executable data analysis pipeline. The components may be part of the data management system, part of another application (invoked through system calls, executing SQL or SAS scripts, etc.). In addition to providing users with a mechanism to define and execute these QoD analytic tasks, our long-term research perspectives are to design and support end-to-end analytic workflows for QoD evaluation, e.g., to provide tools for accessing external resources (data sources or functions), archival of metadata (QoD measures and descriptive metadata), optimizing and monitoring of QoD analytic workflow execution. In this context, we also want to propose various alternatives for assisting the user in the design of QoD analytic workflows depending on resource limitations.

Since QWITH query optimization is our ongoing work, the interesting issues concerning: *i)* the heuristics we've defined for building the algebraic annotated QWITH query trees, *ii)* the techniques for query rewriting, *iii)* the cost model for QWITH query processing, and *iv)* the algorithms we propose for relaxing the constraints of QWITH queries, will remain out of the scope of this paper. These important aspects constitute our immediate perspectives of research and development for improving the QWITH query engine. As our current work, this includes more precisely the elaboration of reasonable solutions for the main following issues: *i)* the on-line computation of analytic functions, *ii)* the checking of dual or overlapping constraints in the sense that they partition the data set into mutual exclusive or inclusive subsets when answering data quality-constrained queries, and *iii)* the automatic relaxation of the constraints (determination of ϵ) when no data satisfies the constraints or none is returned if a non-empty query result is still possible.

6. REFERENCES

[1] J. Akoka, L. Berti-Équille, O. Boucelma, M. Bouzeghoub, I. Comyn-Wattiau, M. Cosquer, V. Goasdoué-Thion, Z. Kedad, S. Kinane, V. Peralta, and S. Sisaid Cherfi. A Framework for Quality Evaluation in Data Integration Systems. In *Proc. of the 9th Intl. Conf. on Enterprise Information Systems, ICEIS 2007*, pages 170–175, Funchal, Madeira, Portugal, June 12-16, 2007.

[2] P. Andritsos, A. Fuxman, and R. J. Miller. Clean Answers over Dirty Databases: A Probabilistic Approach. In *Proc. of the 22nd Intl. Conf. on Data Engineering, ICDE 2006*, page 30, Atlanta, GA, USA, April 3-8, 2006.

[3] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer-Verlag, 2006.

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.

[5] L. Berti-Équille. Quality Awareness for Data Management and Mining. Habilitation à Diriger des Recherches. Université de Rennes 1, France, June 2007. [available online].

[6] M. Bouzeghoub and V. Peralta. A Framework for Analysis of Data Freshness. In *Proc. of the 1st Intl. ACM SIGMOD 2004 Workshop on Information Quality in Information Systems, IQIS 2004*, pages 59–67, Paris, France, June 18, 2004.

[7] S. Bowers and B. Ludäscher. Actor-Oriented Design of Scientific Workflows. In *Proc. of the 24th Intl. Conf. on Conceptual Modeling, ER 2005*, volume 3716 of *Lecture Notes in Computer Science*, pages 369–384, Klagenfurt, Austria, October 24-28, 2005.

[8] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, S. Seltzsam, and K. Stocker. ObjectGlobe: Open Distributed Query Processing Services on the Internet. *IEEE Data Eng. Bull.*, 24(1):64–70, 2001.

[9] N. N. Dalvi and D. Suciu. Management of Probabilistic Data: Foundations and Challenges. In *Proc. of the 26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–12, Beijing, China, June 11-13, 2007.

[10] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.

[11] H. Guo, P.-Å. Larson, and R. Ramakrishnan. Caching with 'Good Enough' Currency, Consistency, and Completeness. In *Proc. of the 31st Intl. Conf. on Very Large Data Bases, VLDB 2005*, pages 457–468, Trondheim, Norway, August 30 - September 2, 2005.

[12] M. A. Hernández and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[13] D. Lembo, M. Lenzerini, and R. Rosati. Source Inconsistency and Incompleteness in Data Integration. In *Proc. of the 9th Intl. Workshop on Knowledge Representation meets Databases, KRDB 2002*, volume 54, Toulouse, France, April 19-21, 2002.

[14] S. E. Mendes-Sampaio, C. Deng, and P. Sampaio. Incorporating the Timeliness Quality Dimension in Internet Query Systems. In *Proc. of the Intl. Workshop on Web Information Systems Engineering, WISE 2005*, pages 53–62, New York, NY, USA, November 20-22 2005.

[15] G. A. Mihaila, L. Raschid, and M.-E. Vidal. Source Selection and Ranking in the WebSemantics Architecture: Using Quality of Data Metadata. *Advances in Computers*, 55:89–119, 2001.

[16] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara. Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS (Demo). In *Proc. of 3rd Biennial Conf. on Innovative Data Systems Research*, pages 269–274, Asilomar, CA, USA, January 7-10, 2007.

[17] F. Naumann, J. C. Freytag, and U. Leser. Completeness of Integrated Information Sources. *Inf. Syst.*, 29(7):583–615, 2004.

[18] F. Naumann, U. Leser, and J. C. Freytag. Quality-Driven Integration of Heterogenous Information Systems. In *Proc. of the 25th Intl. Conf. on Very Large Data Bases, VLDB 1999*, pages 447–458, Edinburgh, Scotland, UK, September 7-10, 1999.

[19] S. Parsons. Current Approaches to Handling Imperfect Information in Data and Knowledge Bases. *IEEE Trans. Knowl. Data Eng.*, 8(3):353–372, 1996.

[20] V. Peralta. *Data Quality Evaluation in Data Integration Systems.* [illegible] Université de Versailles, [illegible] Universidad de la República, Uruguay, 2006.

[21] A. P. Sheth, C. Wood, and V. Kashyap. Q-Data: Using Deductive Database Technology to Improve Data Quality. In *Proc. of the Intl. Workshop on Programming with Logic Databases, ILPS*, pages 23–56, Vancouver, BC, Canada, October 26-29, 1993.

[22] M. Weis, F. Naumann, and F. Brosy. A Duplicate Detection Benchmark for XML (and Relational) Data. In *Proc. of the 3rd Intl. ACM SIGMOD 2006 Workshop on Information Quality in Information Systems, IQIS 2006*, Chicago, IL, USA, June 30, 2006.

[23] W. E. Winkler. Overview of Record Linkage and Current Research Directions. Technical report, Statistical Research Division, U.S. Census Bureau, February 2006.

Toward a Unified Model for Information Quality

C. Batini, D. Barone, F. Cabitza,
G. Ciocca, F. Marini, G. Pasi and R. Schettini
Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli studi di Milano-Bicocca
20126, Viale Sarca 336
Milano, Italy
{batini,barone,cabitza,ciocca,marini,pasi,schettini}@disco.unimib.it

ABSTRACT

We present a model which allows to define in an uniform way information quality dimensions related to heterogeneous types of information, such as structured data managed in data bases, semi-structured and unstructured texts and images. We first define a set of concepts that allow to represent several basic characteristics of such heterogeneous types of information. Then, we introduce a general categorization of quality dimensions and sub-dimensions, which are then specialized to structured data, semi- and unstructured texts and images. In so doing, we provide, to our knowledge, a first attempt to unify the information quality issue for heterogeneous information types.

Keywords

data quality, information quality, structured data, semistructured data, unstructured text, image, quality dimension, quality metrics.

1. INTRODUCTION

In last twenty years the concept of quality in organizations has been declined in several ways both in private consultancy and academic research domains. Quality in an organization relates to the ability of the organization to fulfill the needs and expectations of its customers and users, to create value in an efficient and effective manner, to control and improve the performance of its processes and the outcomes of its services. With the advent of office automation technologies, and of the digitalization of information systems, organizational quality has increasingly been bound and coupled with the quality of the data that an organization produces, holds and consumes to retain and retrieve valuable information. Now more than ever, information is available in different formats, media and resources and it is accessed and exploited through multiple channels. Available information is not only textual, it is also represented by pictures, videos and sound, all contributing to create the information assets of an organization. Since all these kinds of information are intertwined and can refer to the same conceptual entities in complex and complementary ways, the assessment of information quality should take all of them into account, with appropriate dimensions and metrics. In this paper, we address the problem of assessing information quality when information can be carried by either structured data, semi-structured and unstructured texts, or images. To this aim a unified model for information quality is proposed.

The paper is organized as follows. In Section 2 we define the basic concepts on which our model is grounded; in Section 3 we define the quality model. Based on this approach, and due to the multidimensional nature of information quality, in Sections 4, 5 and 6 we address each type of considered information content in a separate way, i.e. structured alphanumeric data, semi- and unstructured texts and images. Section 7 concludes the paper.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

2. INFORMATION QUALITY

Formally, we refer to a *digital information item* as to an atomic information unit in a considered context. A *digital information item* is constituted by a *content*, a *carrier*, and possibly (this is not mandatory) by a *schema* (see Figure 1). A set of digital information items constitutes a *digital information resource*, on which users can rely to get some information. Any digital information resource must be represented in some way, by what we call a *digital information representation*. Common examples of digital information representation are databases, inverted indexes and raster pictures. The *content* of a digital information item is constituted by an *essence* and possibly by some *metadata* and *annotations* associated with it. *Essence* is the real material itself, the result of the creative process, what can be used to produce value or to inform actors and their processes. *Metadata* may be associated with the essence in order to characterize a set of its properties (like the name of the author or the date of creation). *Annotations* constitute any additional and explanatory information that may be associated with the essence. The *content* is then simply the combination of essence, metadata and annotations, i.e., the kernel of information plus what describes and enriches it. While the essence must exist for a content to exist, metadata and annotations can also be absent. The essence of a *content* can be of different types: in this paper, we consider that essence may be expressed in terms of i) *alphanumeric data*, namely symbols, numbers and terms pertaining to a specific domain of values; ii) texts, namely self-explanatory sentences, narra-

tive passages, or any written work; and iii) images, namely, visual content which encodes information either about the geometry of scenes and the properties of the objects located within these scenes [15], or about a visual representation of a non-visual information, like in the case of maps. Alphanumeric and textual content can be further classified with respect to its *level of structuredness*, i.e., as *structured content, semi-structured content and unstructured content*. When considering alphanumeric content, this level of structuredness varies according to the *schema*, e.g., whether data types, domain types and semantic constraints are explicitly expressed or not. When considering texts, the level of structuredness varies according to whether a *schema* exists (as in the case of XML-marked up semi-structured text) or whether a reader can recognize any meaningful structure within the *carrier*.

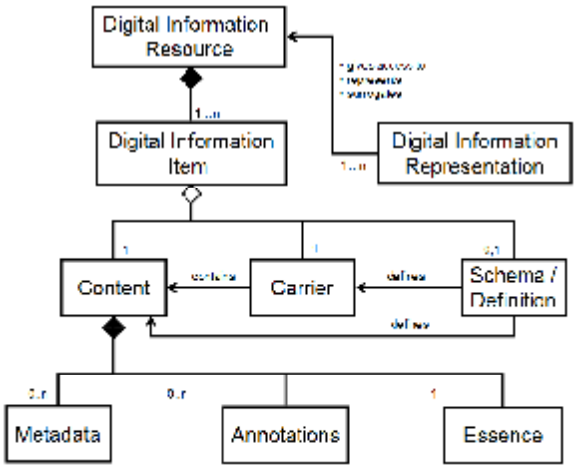


Figure 1: The basic concepts of the paper and their main relationships.

With the term *schema* we denote the way in which the structure of the content and its semantics are explicitly declared in terms of formats, types, constraints and relationships between either the values (e.g., in structured data), fields (e.g., in structured and semi-structured text), sections (e.g., in unstructured but formatted text) or partitions (e.g., in images) of the digital content. The concept of *carrier* of the content of a digital information item is strongly related with the concept of schema.

We introduce an example to clarify the concepts previously defined. Let us consider a digital information item constituted by an XML file (document). The schema of such an information item is constituted by the XML schema. An example of content is the name 'Valentino Rossi' embedded in a tag defining names as alphabetic strings of max 25 characters. In this case the carrier is the tag 'name' defined in the XML file and whose inner value is 'Valentino Rossi'. Metadata and annotations are not mandatory. An example of metadata would be the 'creation date' and 'file dimension' of the above mentioned XML file. An annotation would be any external resource that, after the creation of the XML file, is linked to the XML file for later reference and addition. Readers can also refer to Figure 2 for more examples.

From the above general perspective, any digital information item can be seen as a content, whose characteristics change depending on either the context of use, the representational medium, or the goal of use. In this context, by *Information Quality* (IQ) we refer to the capability of a set of digital information items to meet a set of requirements, either explicitly or implicitly expressed, pertaining to their use in a given context under specific conditions. The quality of a digital information item can be assessed with respect to its content, and according to a specific schema, if any. This means that quality can be assessed with respect to both the essence of an information item, as well as to its metadata and annotations. Hence, the overall quality of a digital information item can be the composition of assessments carried out for different types of content. Moreover, even with respect to content, IQ can be assessed in different ways according to the type of content.

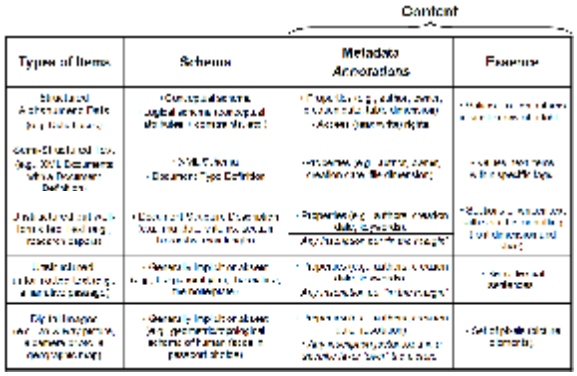


Figure 2: Examples of schema and content in the digital domain.

In this paper, we aim to contribute to the development of a comprehensive approach to IQ modeling of digital information items such as multimedia documents, i.e. documents composed by alphanumeric structured data, text, and images. Hence we aim to define a general model of information quality that can be used to conveniently assess and improve the quality of information produced, used and exchanged in any organizational domain. The proposed model can help analysts and users in keeping track of the quality factors that are of interest in the domain of digital information.

In what follows, more specific definitions will be proposed for each content type in order to characterize the related quality dimensions. To our knowledge, our contribution is a pioneering one, although similar approaches are being carried out by standardizing bodies world-wide. Accordingly, we refer to the comprehensive Data Quality Model that is under development within the ISO project SQuaRE (Software Product Quality Requirements and Evaluation). At the present moment, this model is denoted as ISO/IEC FCD 25012.2 standard. Currently, the development of the ISO/IEC 25012 standard is at the enquiry stage (i.e., its current Draft International Standards is on ballot). In what follows, we will briefly denote this standard as the 'ISO standard'.

In the research literature, IQ is modelled in terms of what are usually called *dimensions* or *characteristics*. All IQ dimensions can be traced back to three macro-categories: *in-*

trinsic, *external* and *contextual* [19]. These approximately correspond to the internal, external and in-use categories of the ISO standard. Intrinsic dimensions depend on the information itself and its schema (if any), irrespectively of where and how it is used. External dimensions depend on the technological environment and tools where/by which the information is used, and on the technological properties of the information system that encompasses it (either at the software or hardware level). Contextual dimensions depend on the actual environmental, organizational and socio-technical context where information is accessed, used and produced.

Dimensions and sub-dimensions related to intrinsic quality provide criteria upon which to guarantee, assess and improve the quality with respect to either the values (i.e., essence) and the schema (if it exists). Aspects related to the external quality regard properties of the components of the system that manages and provides access to information (e.g., the software that indexes and retrieves unstructured text). Contextual dimensions regard the capability of information to enable users to achieve their own objectives, and to add values into their activity in a specific context. Intrinsic and external aspects are usually measured by means of quantitative and objective metrics. Contextual dimensions are usually expressed in terms of the subjective point of view of actual users by their degree of satisfaction with respect to how data fulfill their operative needs.

3. INFORMATION QUALITY DIMENSIONS, SUB-DIMENSIONS AND METRICS

In this section, we present the basic, well established approach we adopt for defining a unified model for the assessment of IQ of digital multimedia information items. Based on this approach, and due to the multidimensional nature of IQ, in Sections 4, 5 and 6 we address each type of considered information content in a separate way, although complying to the same basic model.

As mentioned in Section 2, *quality dimensions* identify significant and possibly inter-dependent aspects concurring to the assessment of information quality. Based on the well established approach to the definition of quality dimensions in the context of IQ, the attributes of digital information items are categorized into main *IQ dimensions*, which are further classified into *IQ sub-dimensions*, in their turn decomposed in lower-level and measurable *IQ metrics*.

We introduce three high-level dimensions, namely *soundness*, *usability* and *portability*; these are called high-level dimensions, to distinguish them from quality sub-dimensions that refine the high-level ones.

Soundness is the characteristic of being free from defects, good, accurate, complete and reliable. This dimension is related to the extent a content can satisfy either some stated or implicit requirements when used under specified conditions. Soundness can be measured either subjectively or objectively, by either comparing the content to some "gold standard" (e.g., the original version of a text, a target image), which is assumed to be perfect and as sound as possible, or by considering no reference but by taking into account a *correlating model* (e.g., a perceptual model of human vision or a set of grammatical and orthographic rules) between a specific sub-dimension and the value of soundness. Soundness can be seen either from an intrinsic perspective, when the requirements are independent of any technological system; or from an external point of view, when the specified conditions of use involve using some actual technological system.

Usability is related to the characteristic of being of some utility to some user. In other words, this dimension is related to the extent a content can satisfy the actual needs of either who (user) or what (process) exploits it. Usability depends both on the system used to either reproduce or access the content, and on the purpose underlying its use, reproduction and access. Usability is related then to the semantic aspects of a content, i.e., what it means, what objects of interest it represents for a specific aim. Usability encompasses characteristics that pertain to both external and contextual aspects. That is, this is a dimension the value of which depends on the technological layer that makes data usable, as well as on the context of production/use and re-use of the content. Accordingly, usability is usually measured in terms of subjective and user-centered assessments, although also objective measures could be related to usability under simple assumptions (e.g., the more a content is used, in terms of number of accesses and time of use, the more it is usable).

Portability is related to the extent the content is useful also outside the original context of production and use, i.e., the extent to which the content can satisfy the needs of users across specific boundaries, such as cultural, professional, organizational and geographic boundaries. Portability depends on the system of transmission, the interoperability platform which transports, translates and conveys content across a distributed environment. Portability, like usability, can be considered from either an external and contextual point of view. Portability is usually measured in terms of subjective and user-centered assessments. In the following, being portability a technological issue, we will not address it.

3.1 Quality Sub-dimensions

Several sub-dimensions can be identified in order to further refine the concept of information quality with respect to soundness and usability.

The quality of each type of content (alphanumeric structured data, text, image) can be characterized in terms of sub-dimensions that are peculiar to the considered type of content. Their names and their definition can also differ with respect to the application domain (e.g., usability of images in medical imaging differs from usability of images for e-commerce applications).

The most frequently mentioned sub-dimensions are *accuracy*, *consistency*, *completeness*, *accessibility*, and *time-related* dimensions, like *timeliness* and *currency*, since they are likely significant in every application domain. A number of these core sub-dimensions can be applied to both text-based content and images: to this aim, the definitions that have been given in the literature for a specific type of data must be adjusted to be sufficiently generic to be applicable also to the other kinds of content.

In the following, we focus on *accuracy*, *completeness* and *readability*, and provide general definitions according to the domain of analysis.

Defining a quality dimension implies to associate it with a set of related metrics. By the term *metrics*, we refer to the definitions given within both the ISO standard 9126-1 and the ISM3 framework [2], and hence we refer to a set of elements encompassing both a *measurement procedure*,

i.e., an algorithm that takes the element to measure and associates it with a measure (be it ordinal or interval value), and a proper *unit of measure*, i.e., the domain of values returned by the measuring procedure. In general, several metrics can be associated with each quality dimension.

Indeed, a quality dimensions can be described and characterized both at a semantic and a syntactic level. These two levels are obviously related but lead to slightly different definitions and ways to assess and measure a given dimension of a given information representation. We consider as *semantic* the level of description of a dimension that characterizes a representation with respect to a phenomenon in the represented reality of interest. We define *syntactic* the level at which a dimension characterizes a representation with respect to a reference representation (value or structure) in the same domain of representation. This twofold characterization of the concept of quality dimension is common to all types of information. For instance, an image can be assessed either intrinsically, as a stand-alone entity that can not be matched to any reference image or sound, or with respect to a reality that such content is intended to represent, e.g., the face of a person. The same can be said for a database record of a table representing the employees of an organization: its accuracy (as composition of the accuracy of each item) can be assessed both with respect to a reference vocabulary (syntactic accuracy) and with respect to the actual community of employees of that organization. The two assessments of the same entity (the record) can be quite different from each other, as in the case where, e.g., 'John' is an accurate item with respect to the set of all English names but no John is actually working at that particular organization.

4. QUALITY DIMENSIONS FOR ALPHANUMERIC STRUCTURED DATA

We notice that no general agreement exists either on which set of dimensions defines the quality of alphanumeric data, or on the exact meaning of each dimension. For a comprehensive discussion on this issue see [5].

Several definitions are provided for the term *accuracy*. [21] define accuracy as "the extent to which data are correct, reliable and certified"'. [9] specify that data are accurate when the data values stored in the database correspond to real-world values. In [17], accuracy is defined as a measure of the proximity of a data value v to some other value v' that is considered correct. In general, two types of accuracy can be distinguished, syntactic and semantic, which we adopt in the following.

Syntactic accuracy is the closeness of a value v to the elements of the corresponding definition domain D. In syntactic accuracy, we are not interested in comparing v with its real-world value v'; rather, we are interested in checking whether v is any one of the values in D, or how close it is to values in D. For example, v = *'Jean'* is considered syntactically accurate even if v' = *'John'*.

Semantic accuracy is the closeness of value v to the corresponding true (real-world) value v'. While it is reasonable to measure syntactic accuracy using a distance function, semantic accuracy is measured with a boolean <yes, no> or a <correct, not correct> domain. Consequently, semantic accuracy coincides with the concept of *correctness*. In general, techniques assessing and improving semantic accuracy are considerably more complex than techniques addressing syntactic accuracy.

Completeness is defined as the degree to which a given data collection includes the data describing the corresponding set of real-world objects.

In the research area of relational databases, completeness can be referred to any type of structure in the model, resulting in:

- a *value completeness*, to capture the presence of null values for some fields of a tuple;
- a *tuple completeness*, to characterize the completeness of a tuple with respect to the values of all its fields;
- an *attribute completeness*, to measure the number of null values of a specific attribute in a relation;
- a *relation completeness*, to capture the presence of null values in a whole relation.

In all these cases, completeness is related to the meaning of *null* values defined in the model. A null value has the general meaning of *missing value*, i.e. a value that exists in the real world but is not available in a data collection. In order to characterize completeness, it is important to understand why the value is missing. A value can be missing either because it exists, but is not known, or because it does not exist, or because it is not known whether it exists (see [3]).

Let us consider the table reported in Figure 3, with attributes `Name`, `Surname`, `BirthDate`, and `Email`. If the person represented by tuple 2 has no email, tuple 2 is complete. If the person represented by tuple 3 has an e-mail, but its value is not known then tuple 3 presents an incompleteness. Finally, if it is not known whether the person represented by tuple 4 has an e-mail or not, incompleteness may or may not occur, according to the two cases.

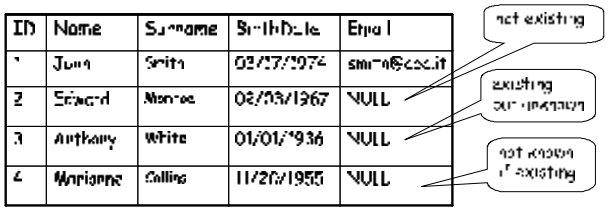


Figure 3: Null values and data completeness

In logical models for databases, such as the relational model, there are two different assumptions on the completeness of data represented in a relation instance `r`. The *closed world assumption* (CWA) states that only the values actually present in a relational table `r`, and no other values represent facts of the real world. In the *open world assumption* (OWA) we can state neither the truth nor the falsity of facts not represented in the tuples of `r`.

From the four possible combinations emerging from (i) considering or not considering null values, and (ii) OWA and CWA, we consider the two most interesting cases:

1. model without null values with OWA;
2. model with null values with CWA.

In a model without null values with OWA, in order to characterize completeness we need to introduce the concept of *reference relation*. Given the relation `r`, the reference relation of `r`, called `ref(r)`, is the relation containing all the tuples that satisfy the relational schema of `r`, i.e., that represent objects of the real world that constitute the present true extension of the schema.

On the basis of the reference relation, the completeness of a relation `r` is measured in a model without null values as the fraction of tuples actually represented in the relation `r`, namely, its size with respect to the total number of tuples in `ref(r)`:

$$C(r) = \frac{|r|}{|ref(r)|}$$

In the model with null values with CWA, specific definitions for completeness can be provided by considering the granularity of the model elements, i.e., value, tuple, attribute and relations.

Concerning *readability*, intuitively a database, or also, its schema, is readable whenever it represents the meaning of the reality represented by the schema in a clear way for its intended use. This simple, qualitative definition is not easy to translate in a more formal way, since the evaluation expressed by the word *clearly* conveys elements of subjectivity. In models, such as the Entity Relationship model, that provide a graphical representation of the schema, called *diagram*, readability concerns both the diagram and the schema itself.

Another aspect related to readability is the property that every aspect of the real world is represented by a specific single database structure; this characteristics results in the relational model in the concept of normalization. The property of *normalization* has been deeply investigated, especially in the relational model, although it expresses a model-independent, general property of schemas. In the relational model, normalization is strictly related to the structure of functional dependencies. Several degrees of normalization have been defined in the relational model, such as first, second, third, Boyce Codd, fourth, and other normal forms. The most popular and intuitive normal form is the *Boyce Codd normal form, BCNF* (see [3]). A relation schema `R` is in BCNF if for every non trivial functional dependency `X -> Y` defined on `R`, `X` contains a key `K` of `R`, i.e., `X` is a superkey of `R`. The interpretation of BCNF is that the relational schema represents a unique concept, with which all nontrivial functional dependencies are homogeneously associated, and whose properties are represented by all non-key attributes. For more details on the BCNF and other normal forms, see [3] and [10].

5. QUALITY DIMENSIONS FOR TEXTUAL INFORMATION

5.1 Dimensions and metrics for Textual Information

In this section, we define a set of quality dimensions that can be applied to describe relevant aspects of textual information, be it either semistructured – as in an XML file – or unstructured – as in any narrative text. For each dimension, we provide an operative definition, and for some of them we either suggest or define a quantitative metric. To this aim, we consider the main structures conveying textual information. The smallest data structure (information unit) that we consider significant in the domain of textual information is the *word*. Words can be considered atomic elements to our practical purposes and can be either *simple* or *compound*: in the latter case, two or, less likely, more words are considered as an unity. Words are usually separated by spaces in texts; blanks can be considered as the simplest delimiters of words. The first level of words aggregation that we take into consideration is the *sentence* level. Sentences are grammatical units of one or more words, bearing syntactic relation to the words that precede or follow it. Sentences are usually separated by punctuation symbols, such as periods, semicolons and other terminal punctuation marks. We are not concerned with the meaning of sentences, nor with its either intrinsic or contextual quality; this is the object of study of the discipline called *linguistics*, whose concern is quite far from the scope of this paper. The next structure to which we apply the typical quality dimensions seen in the previous section is *text*. For the notion of text, we assume the common sense definition of set of sentences that can be considered as unitary as a single and meaningful body of matter in a manuscript, book, newspaper, etc. The next and last level we consider is that of *collection of texts*, i.e., a group of single texts that are gathered into one location, for some purpose or as a result of some process. With reference to Figure 1, we consider texts represented in any digital form as our digital information items (in which words are the atomic information units), and collections of texts as our digital information resources.

5.2 Specific dimensions and Related Metrics for Textual Information

According to [16], *accuracy* is the foundation measure of the quality of data. Moreover, its impact on the overall quality is significant, since if one makes grammatical mistakes, other aspects like conciseness and elegancy are of little importance. As in the case of structured alphanumeric data – as seen in Section 4 – the concept of accuracy of an information item can be declined in terms of both a syntactic accuracy and a semantic accuracy. In the case of texts, we adapt the definition provided for alphanumeric data by first applying it to atomic information units, and then generalizing it to a whole information item (text). This means that if a word that is used in a text belongs to a reference dictionary or word list, it may be considered accurate, leaving its meaning out of consideration. Syntactic accuracy is then first assessed with respect to single words, then generalized to texts and their collections, since its conceptual assessment is scalable (e.g., by simple composition or aggregation) and its factual assessment can be carried out in quantitatively and automatic manner. Conversely, semantic accuracy regards how well a word, a text or passage, describes, i.e., is faithful to a real situation or phenomenon of the reality of interest. In this case, we should take into account the meaning of words, as well as their relationships, and also the reality they intend to represent accurately. This is an aspect that is very difficult to assess either automatically or quantitatively. That said, we propose the following definitions of accuracy related to textual information.

Syntactic Accuracy is the degree to which the reported information item is in conformance with the elements of a reference language vocabulary V. Intuitively, a quantitative metrics to assess syntactic accuracy of a textual information

item should count the number of words that are correct from the spelling point of view, and compare such number with the total number of the words contained in the text.

Semantic accuracy is the degree to which the reported information value is in conformance with the true or accepted value [6]. In this case, subjective metrics must be adopted as the only feasible approach. These metrics can differ a lot in the domain design and measure method, but they are basically based on the qualitative assessment carried out by a group of human readers, which are either expert of the subject the text is about, or not. Another approach could consider how many *specific words* are used in a sentence (or text, collection). Yet, it is not a trivial task to consider how specific a word is. Intuitively, a word is specific if, in a given thesaurus no other word specializes its intended meaning. For instance, if in a sentence it is used the term 'thing' to refer some object, e.g., a table, and in another describing the same situation it is used the more specific term 'table' or even 'pool-table', the latter sentence would be considered more accurate than the former. The metrics adopting this approach have to rely on a semantic lexicon for the given language used in the text, which provides for each noun, verb, adjective and adverb their synonym sets, each representing a single underlying lexical concept (e.g., such as WordNet).

Completeness refers to the extent an information item/ resource has all parts or elements that are needed for a certain task or with respect to a certain schema. These two cases can be somehow related to the case of semantic and syntactic completeness, respectively. In regard to syntactic completeness, in semistructured textual resources, such as XML files, a schema exists and is explicit. In this case, the metrics to assess completeness can be borrowed from the field of alphanumeric structured data, and be based on the simple count of fields (tags) that are filled in, with respect to the total number of tags of the whole sentence, text or collection of texts. On the other hand, in case of unstructured texts, completeness can be assessed with respect to underlying requirements or explicit constraints. For instance, an abstract of a conference paper is usually something that is considered either complete or not according to a word limit fixed by the organizing committee. The convention that holds here is that the word limit is to be intended not as an upper limit, but rather as the advisable number of words needed to let the reviewers and readers understand what the paper is about. In such cases, completeness can be expressed in terms of a ratio between the number of terms used and the number of terms considered optimal for a given task (mind that in this case, also values higher than one are possible, although not always desirable).

A similar approach can be adopted also regarding the distinction between completeness under the closed world assumption and the open world assumption. In the former case, the only way to assess the completeness of a textual information item is to count the number of omissis, i.e., terms of the text deliberately left out and substituted by some place-holder (e.g., three dots, a black label). Completeness would be assessed as the ratio between the number of omissis present in a sentence (or text, collection) and the sum between this number and the overall word count. In the open world assumption, only qualitative and subjective metrics can be adopted, as in the case of semantic accuracy. In this case a panel of human experts should be consulted in order to understand whether the read text describes the object of description in a complete manner or not.

Readability is the extent to which a text is easy to be read and understood for its targeted audience. To assess readability we may use, among others, the Gunning fog index [12], which measures the level of reading difficulty of any documents. The main idea of this method is that the higher the complexity of each sentence and the bigger the words used in it, the higher is difficulty to read the text. The resulting number is an indicator of the number of years of education a reader requires to easily understand the text at a first reading. The "standard" score is 7 or 8; and a text with score above 12 is considered hard to read for most people. The Gunning fog index may be calculated by the following algorithm:

1. Select a short passage of the text (usually around 100 words) and count the number of words. For a lengthy document, select several passages and average the Fog index.
2. Count the number of sentences.
3. Count the number of complex words, i.e. words with three or more syllables, not including proper nouns (for example, 'Frederick'), compound words like "newspaper", or common suffixes such as -es, -ed, or -ing as a syllable, or familiar jargon.
4. Calculate the average sentence length (i.e., divide the number of words by the number of sentences).
5. Calculate the percentage of complex words.
6. Add the average sentence length and the percentage of complex words, and multiply the result by 0.4

The complete formula is as follows:

$$0.4 * \left(\left(\frac{\text{words}}{\text{sentence}} \right) + 100 \left(\frac{\text{complex words}}{\text{words}} \right) \right) \quad (1)$$

6. QUALITY DIMENSIONS FOR IMAGES

6.1 Dimensions and metrics for images

In general, image quality dimensions can be assessed either for an image seen in isolation (e.g. readability) or for an image seen together with the reference (e.g. semantic accuracy). Depending on the image data and application, quality dimension assessment can be done by psychological experiments involving human observers or computing suitable metrics directly from the digital image, these metrics can be eventually combined as proposed in [4] and [11]. Standard psychophysical scaling tools for measuring subjective image quality are available and described in specific standards, such as ITU-R BT.500-11 [20], [13]. The involvement of real people that view the images to assess their quality requires that all the factors that influence perception are taken into account and strict protocols are adopted. Subjective image quality assessment necessarily involves taking into account both the Human Vision System characteristics and the image rendering procedure, subjects' characteristics and the perceptual task. Subjective image quality may be assessed only when the image is viewed by an observer. Objective image quality measures not requiring human interaction can be broadly classified in two classes: signal-based

Figure 4: Low syntactic accuracy.

metrics, such as the root mean square error; and perceptually based metrics, which relate image signals to perceived quality. These metrics include simplified models of the human vision system, such as the visible differences predictor model [8].

6.2 Specific Dimensions and Related Metrics for Images

Following [5], we can define the *syntactic accuracy* of an image as its closeness to an image in the chosen application domain. For example, if the application is a biometric authentication system based on face detection and recognition, any image that does not contain a detectable/recognizable face should be considered outside the application domain and thus discarded. Syntactic accuracy can be assessed either by visual inspection or by pattern recognition techniques [14]. An example is shown in Figure 4, related to a face-based biometric recognition system. In such systems the detection of a face in an image can be used to discriminate between syntactically accurate images (first three images from the left) and other images (right). In this case, automatic determination of the syntactic accuracy of the images can be accomplished using a face detection algorithm [22]. In the rightmost image the face recognition fails.

Image semantic accuracy can be defined as the degree of matching (fidelity) of the digital image with respect to the corresponding (measured) external reference in the reality of interest, i.e. the original scene or source data we want to represent. If the image is synthesized from non-image data, its semantic accuracy is obviously related to the semantic accuracy of the data itself. Figure 5 shows an example of images with different levels of semantic accuracy (fidelity) with respect to the reference image. The background of the middle image is noisy while the color and the logo text of the left image are wrong. In this figure, the middle and right images are examples of low semantic accuracy with respect to the reference image (depicted on the left).

In some cases, semantic accuracy could be assessed by a human viewer without requiring the availability of the external reference. In these cases we can define a "reduced" semantic accuracy as the degree of apparent match of the image with the viewer's internal references [15]. Examples of image requiring a high degree of naturalness are those seen on journals. Naturalness plays a fundamental role when the image to be evaluated does not exist in the reality, such as in virtual reality domains. In Figure 6 we see two examples of images, that, despite being faithful with respect to the original scene (left) or the source data (right), lack of naturalness.

In regards to completeness, we can adapt the definition of completeness provided for alphanumeric data and texts as follows: a digital image is *complete* if it depicts all the

Figure 5: Low semantic accuracy.

Figure 6: Lack of naturalness.

information that it must convey. There are several causes for the lack of completeness and they can be related to i) acquisition or production process that generate the image (e.g. Figure 7) or to ii) the represented phenomenon (real or synthesized) being intrinsically incomplete (e.g. Figure 8).

Figure 7 shows a subject acquired with a 3D laser scanner. It can be seen that the image exhibits several holes where the scanner has been not able to correctly acquire the model surface. The 3D representation of the model is thus incomplete.

Figure 8 shows a partially occluded face. The incompleteness of this image is related to the incompleteness of the information in the acquired scene. In this case the image depicts the entire scene's visible information but the scarf

Figure 7: First example of incompleteness.

Figure 8: Second example of incompleteness.

occludes other information concerning the traits of the subject's face.

The definition of completeness given above can be applied both to a single image as well as to a set of images intended to capture a single phenomenon. For example, the acquisition of a large painting at high resolution is usually done by topologically subdividing it into small tiles that are individually acquired at the required resolution. An example of this acquisition technique can be found in [1] where the mural painting "The Last Supper" of Leonardo da Vinci was acquired using 1,677 tiles resulting in a virtual image of size 172,181 x 93,611 pixels. This set of tiles must be complete to be properly combined to provide the whole painting.

We finally define *readability* of an image as the lack of distortions or artifacts that reduce the accessibility of its information contents. Some of the most frequent artifacts are: *blurriness*, *graininess*, *blockiness*, *lack of contrast* and *lack of saturation*. To detail each of these terms falls out of the scope of the paper. In Figure 9, we provide some examples of artifacts that may reduce the readability of an image. The readability can be assessed either directly by human visual inspection or indirectly by automatically estimating the presence and the strengths of these artifacts. The estimation's process requires the modeling of the visual effects that an artifact produces on the image in terms of distortions of low level visual features, such as color, edge, texture.

7. CONCLUSIONS AND FUTURE WORK

In this paper we made a first attempt to establish a unified approach to information quality for heterogeneous types of information. We have focused on three quality dimensions, namely, accuracy, completeness and readability. Much work has to be done to formally express such model in terms of a complete classification of quality dimensions that encompass all types of qualities mentioned in the literature [5], and to extend the approach to other relevant information types such as maps [18] and sounds [7].

8. REFERENCES

[1] Available on line: `www.haltadefinizione.com/en/cenacolo/index.asp`.

[2] V. Aceituno. ISM: Information security management maturity model - handbook. Technical report, ISM3 Consortium, 2007.

[3] P. Atzeni and V. de Antonellis. *Relational Database Theory*. The Benjamin /Cummings Publishing Company, Inc., 1993.

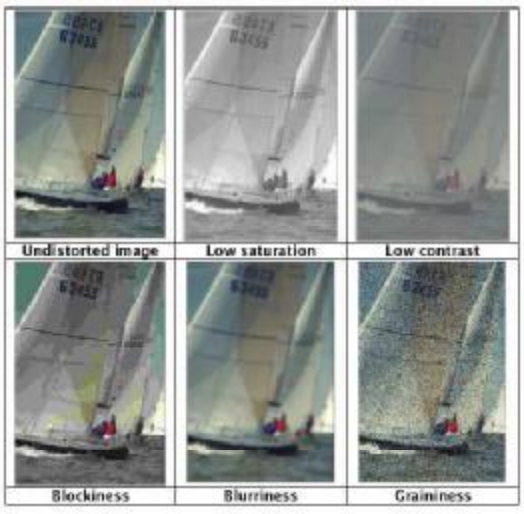


Figure 9: Low readability.

[4] C. J. Bartleson. The combined influence of sharpness and graininess on the quality of colour prints. *Journal Photog. Science*, 1982.

[5] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies, Techniques*. Springer Verlag, 2006.

[6] D. Becker, W. Mcmullen, and K. Hetherington. A flexible and generic data quality metamodel. *International Conference on Information Quality*, 2007.

[7] J. Blauert and U. Jekosch. *Sound-Quality Evaluation - A Multi-Layered Problem - EAA-Tutorium on Aurally Adequate Sound-Quality Evaluation*. EAA, Antwerp, Netherlands, March 1996.

[8] S. Daly. The visible difference predictor: an algorithm for the assessment of image fidelity. *Digital images and Human vision*, pages 179–206, 1992.

[9] P. H. D.P. Ballou. Modeling data and process quality in multi-input, multi-output information systems. *Management Science*, 31(2), 1985.

[10] R. Elmasri and S. Navathe. *Foundamentals of database systems, Fifth Edition*. Addison-Wesley Publishing Company, 1994.

[11] P. Engeldrum. Psychometric Scaling: Avoiding the Pitfalls and Hazards. pages 101–107, 2001.

[12] R. Gunning. *The Technique of Clear Writing*. McGraw Hill, New York, NY, 1952.

[13] ITU. Methodology for the Subjective Assessment of the Quality for Television Pictures. ITU-R Rec. BT. 500-11, 2002.

[14] R. Jain, S. Antani, and R. Kasturi. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition*, 35(4):945–965, 2002.

[15] T. J. W. M. Janssen and F. J. J. Blommaert. Predicting the usefulness and naturalness of color reproductions. *Journal of Imaging Science and*

Technology, 44:93–104, 2000.

[16] J. Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[17] T. Redman. *Data Quality for the Information Age*. Artech House, 1996.

[18] Shekhar, Shashi, Xiong, and Hui, editors. *Encyclopedia of GIS*. Springer Verlag, 2008.

[19] D. M. Strong, Y. W. Lee, and R. Y. Wang. Data quality in context. *Communications of ACM*, 40(5):103–110, 1997.

[20] L. Thurstone. A law of comparative judgement. *Psychological Review*, 34:273–286, 1927.

[21] R. Wang and D. Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4), 1996.

[22] M.-H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

Model-Driven Component Generation for Families of Completeness Measures

Nurul Akmar Emran
School of Computer Science
University of Manchester
Oxford Rd,Manchester,UK
emrann@cs.man.ac.uk

Suzanne Embury
School of Computer Science
University of Manchester
Oxford Rd,Manchester,UK
sembury@cs.man.ac.uk

Paolo Missier
School of Computer Science
University of Manchester
Oxford Rd,Manchester,UK
pmissier@cs.man.ac.uk

ABSTRACT

Completeness is a well-understood dimension of data quality. In particular, measures of coverage can be used to assess the completeness of a data source, relative to some *universe*, for instance a collection of reference databases. We observe that this definition is inherently and implicitly multi-dimensional: in principle, one can compute measures of coverage that are expressed as a combination of subset of the attributes in the data source schema. This generalization can be useful in several application domains, notably in the life sciences. This leads to the idea of declaratively specifying families of completeness measures that users can choose from. Furthermore, individuals in the family can be specified as OLAP-type queries on a dimensional schema. In this paper we describe an initial data architecture to support and validate the idea, and show how dimensional completeness measures can be supported in practice by extending the Quality View model [11].

1. INTRODUCTION

Of the various forms of information quality (IQ) identified in the literature, completeness has been one of the best studied and one of the most precisely defined. In particular, it has been recognised that completeness is a complex quality, and that many different forms can be envisaged. A common distinction, for example, is to separate completeness of a data set in terms of the number of individuals represented (relative to the "true" population modelled by the data) from completeness in terms of the amount of data recorded about each individual (relative to the full amount of information that could possibly be collected about the individual). These forms of completeness are commonly called *coverage* and *density* (after Naumann *et al.* [15]), and each demands a quite different approach to measurement.

We have been undertaking a study of the requirements for measuring IQ in various e-Science domains, with the aim of identifying patterns of IQ measure that are widely applicable, and which can be tailored for use in specific applications [11]. Recently, we have turned our attention to the issue of information completeness, which is of particular importance in many e-Science applications. A typical format for *in silico* experiments in e-Science is first to identify one or more public repositories which can provide the input to the experiment, to select from and clean up the data they provide, and then to execute the experiment (perhaps described in the form of a workflow [8]) over the selected data sets. As we shall show later in this paper, the completeness of the data sets over which the experiments run can have significant effects on the correctness or usefulness of the results.

In order to elicit more precise requirements for completeness measurements in e-Science, we have looked specifically at the domain of SNP databases. A SNP (pronounced *snip*) is a single nucleotide polymorphism; that is, a change in a single base of a gene that is observed in a significantly large proportion of the population of a species to represent a specific trait within that population (rather than just a random mutation). Taken *en masse*, SNPs represent the genetic diversity of a species, and so are vital in helping to map phenotypic differences (such as susceptibility or resistance to specific diseases) to their corresponding genetic differences. Because of this, SNPs are typically used in comparative studies of large sections of a species' genome, and as such are particularly sensitive to completeness issues in the underlying data sets.

Our work has revealed a surprising diversity in completeness requirements, even within the standard coverage/density classification found in the literature. Rather than a single generic completeness measure, relative to the main population being accessed in an application, SNP scientists instead are concerned with the completeness of the data relative to certain specific *dimensions*. For example, for some kinds of SNP study, it is important that SNPs for a specific set of strains of interest are included in the underlying data sets. For other applications, strains are not relevant; instead, coverage of certain regions of certain chromosomes is more important. Overall, we have identified a total of 20 dimensions in SNP data that might be important in assessing the completeness of SNP data sets for various applications.

Each such dimension represents a specific completeness measure, for which a "universe" of values must be computationally accessible. Similarly, each combination of dimensions represents a completeness measure that may be of interest to some scientist. In other words, within a domain, there exists not one completeness measure, but a whole family of measures, defined by the dimensions of importance within the domain. Given this diversity, the following questions arise:

- How can a specific completeness measure, i.e. a specific element of the completeness family for the domain, be rapidly and conveniently specified to support a particular application?
- What software infrastructure is required to support the efficient measurement of data relative to any of the possible completeness measures belonging to a family?

In this paper, we report on the results of our initial exploration of these questions. We first assess the literature on information completeness and draw from this previous work some general characteristics of completeness measures (Section 2). We then examine the specific completeness requirements found in the SNP domain, and motivate the concept of dimensional completeness families proposed in this paper (Section 3). From this, we propose a model for dimensional completeness families (Section 4) and consider the software infrastructure that is needed to support completeness families expressed in this model and how far it can be automatically generated from the model (Section 5). Finally, we conclude (Section 6).

2. MEASURES OF COMPLETENESS

Completeness of information is included as a key dimension of IQ in all the major taxonomies proposed in the literature (e.g. [1, 3, 17, 22]). Unlike many other measures, completeness is conceptually very simple: completeness is typically defined as a ratio of the size of a data set of interest relative to the size of the "complete" data set. This "complete" data set (which, in this paper, we term the *universe*) may refer to the state of the real world (e.g. the number of genes actually present in the human genome) or to some stored data set that is believed to be a good approximation of the real world (e.g. the GenBank database[1], which records details of the majority of well-established, experimentally determined human genes, i.e., the set of known genes). Completeness measures using the former kind of universe are sometimes referred to as *absolute completeness* measures (with a similar concept found in the literature as the completeness approximation against a reference relation [20]), while the latter are referred to as *relative completeness*.

In specifying a particular completeness measure, therefore, our two main tasks are to define how we measure the size of a data set, and to identify the contents or characteristics of the universe against which completeness should be assessed. Note that the size measure must be applicable to both the data set under study, and to the universe, in order to produce a meaningful ratio of the two quantities (or else two separate but comparable measures must be calculated).

Two main approaches to measuring the size of a data set can be seen in the literature: counting the number of individuals (given the name *coverage* by Naumann *et al.* [15]) and assessing the amount of information available about each recorded individual (given the name *density* by Naumann *et al.* [15]). Both are useful in some situations, and less useful in others. And both bring their own specific challenges in terms of forming precise and automatable quality measures. For example, obtaining an accurate count of the number of individuals recorded in a data set is complicated by the need to deal with duplicates, while assessing the amount of information recorded about one individual is complicated by issues regarding precision of different data representations.

Ballou and Pazer, for example, discuss the difficulties of assessing the relative information content of the temperature values "below freezing" and "20F" [2]. Clearly the former is less complete than the latter, but precisely how much less complete is difficult to measure precisely. Because of this difficulty, current density measures distinguish only between null and non-null values in terms of information content. For example, Scannapieco and Batini proposed a hierarchy of density measures based upon counting the non-null values in data sets at the level of a tuple, a column or a complete relations [20]. Similar measures were also defined by Naumann *et al.* [15], Martinez and Hammer [10] and Motro and Rakov [14]. Such measures are meaningful in contexts where null has a clear and unambiguous meaning - such as in integrated data sets where null values have been inserted whenever the underlying data sources did not contain the information needed to fully populate the attributes of the integrated data. They are less useful in contexts where null has other meanings beyond (and including) "unknown". For example, in many relational data models, null is also used to indicate that a value is inapplicable or empty, as well as when its value is unknown. In such cases, a plain count of nulls will not give a reliable indication of the data completeness.

Even aside from the issue of deduplication/object identity, the specification of universes for completeness measures is a challenging task. Ideally, of course, we would assess completeness against the state of the real world, but this is either impractical or impossible in the vast majority of cases. It is necessary, therefore, to define some proxy for the state of the real world that exists in the technical world and that is sufficiently representative of the real world state to allow meaningful completeness assessments to be made. Two main approaches to universe specification can be distinguished in the literature: virtual (or intensional) and materialised (or extensional).

Virtual universe specifications are defined using rules that describe the relevant characteristics of the complete data set without enumerating it in full. These specifications may be stated explicitly, as part of the measure, or may be implicitly assumed. For example, forms of density measures that are based on counting nulls, such as that of Scannapieco and Batini described above [20], are based on an implicit virtual specification of the universe. In these measures, the universe is assumed to have the same number of individuals and attributes as in the data set being measured, but with the additional knowledge that each attribute is also non-null. Although this specification of the universe does not tell us exactly what values are held by each tuple attribute, it gives enough information for us to be able to assess the number of missing values.

An example of an explicit virtual universe specification, this time in terms of a coverage measure, was given in an early paper on completeness by Motro [13]. In this work, Motro proposed that the data administrator for a data set could give rules describing the set of individuals in the real world that. An example based on a flight information domain used by Motro is the statement that there is one flight daily between Los Angeles and New York. A database of flights can be checked for completeness relative to this portion of the real world, even though the exact set of flights actually scheduled by this airline is unknown. Another example of this kind of universe specification is given by Batini and Scannapieco in the context of public administration databases [3]. If the approximate population of a city is known, then the completeness of a database of residents of that city can be estimated based on this value alone.

Such virtual universe specifications are attractive because of their concise nature and the ease of specification and use. However, there are some obvious limitations. The most ob-

[1] http://www.ncbi.nlm.nih.gov/Genbank/GenbankOverview.html

vious of these is the problem of false positives in the data set, which cannot always be detected by these measures and which can therefore distort the completeness results. For example, the residents database just mentioned might also contain the names of many people who no longer live in the city, and who are not included in the population estimate. The virtual universe specification does not allow us to rule such residents out of the calculation. A further issue is that of maintenance of the rules defining the universe. If these rules do not accurately describe the real world (either because of a change in the world or because they were incorrectly specified) then the completeness measures that result will be of doubtful value.

It is also the case that many (most?) universes cannot be specified in this intensional manner, either because of the complexity of their semantics or because they contain data describing collections of natural kinds that cannot be encapsulated by neat rules. The only way to describe the set of genes in the human genome (whether actual or as currently known), for example, is to list them in terms of the artificial names given to them by scientists. For these kinds of domains, it is necessary to define the universe extensionally, by enumerating its members.

In cases where a single database or resource exists that is a close approximation of the real world, then this can be used as a proxy universe and completeness of other data sets can be measured against it [20]. For example, for practical purposes, the GenBank database referred to earlier is a good proxy universe for the set of known genes (and even for the set of genes that actually exist, given the extreme difficulty of obtaining this information). Completeness of any database of genes can be reliably assessed against its contents.

The number of domains where a reliable single reference resource for completeness exists are small, however. An alternative approach is to construct a universe from the total amount of information that is known; i.e., to construct a single reference set by integrating the multiple sources that are available. Naumann *et al.* made use of this idea in their proposal for a coverage measure based on the idea of a *universal relation* [15]. Assuming a LAV integration approach, these authors derive the universal relation by creating a single relation containing the outer join of all the views exported by the individual sources[2]. This universal relation is then used to assess the completeness of the individual sources, by determining what proportion of the tuples in the full universal relation are supplied by the source being assessed. A universe of this kind will give accurate completeness values in situations where the underlying sources by and large represent a horizontal decomposition of the data being integrated (i.e., where there is an approximate one-to-one correspondence between tuples in the universal relation and individuals in the real world set). In other cases, where (for example) the underlying source views represent a vertical or mixed decomposition of the real world information, the number of tuples in the universal relation will be artificially inflated and its tuples may not correspond to meaningful real world entities. This would occur, for example, whenever foreign keys within the views represent one-to-many relationships of high cardinality. In such cases, some other form of universe specification must be found.

One advantage of materialised universes is that they have the potential to handle false positives correctly, so that they do not distort the completeness measure. However, this is only the case when the data set under study is not part of the set from which the universe is computed are under study. A key (though often unstated) assumption behind this "integration" approach to universe formation is that the sources that make up the universe are accurate (that is, they do not themselves contain false positives). In some domains, it may be easier to manually create and maintain a single reference source for completeness measurement, rather than attempting to guarantee the accuracy of all the sources from which the universe might otherwise be automatically constructed.

Having examined the arsenal of completeness measures proposed to date within the literature, in the next section we will consider how well the measures match up to the requirements for completeness assessment with an example e-Science domain.

3. COMPLETENESS IN SNP DATABASES: A MOTIVATING EXAMPLE

3.1 Overview

One of the primary concerns of biologists is to increase our understanding of the relationship between the genes that an organism acquires from its parents (i.e., its genotype), and the structural and behavioural characteristics it will display as a result (i.e., its phenotype). For example, eye colour in humans (phenotype) is known to be governed by a group of genes, including EYCL1, EYCL2 and EYCL3. Discovering such relationships is challenging, especially for phenotypic traits that are qualitative rather that quantitative, such as those governing susceptibility or resistance to various diseases, and which are governed by a complex interplay of many genes spread across disparate locations in the organism's full genetic sequence.

Single nucleotide polymorphisms (SNPs for short, pronounced "snips") are a specific type of genetic variation that can be used to support many forms of analysis aimed at generating or testing hypotheses relating to such relationships. A SNP is a variation in a single nucleotide base (and therefore in a single gene) that is seen in at least 1% of the population of the species under study [5]. SNPs are significant because they can highlight potential candidate genes for specific phenotypic behaviours, traits or variation studies. The underlying assumption behind many SNP studies is that the genetic factors (i.e., variation) that contribute to an increased risk for a particular condition or disease should be detected at higher rates in the population which exhibits the condition compared to the population which does not [5]. To give a small and unrealistic example of this, suppose one mouse strain is likely to develop a particular congenital defect while another strain is not. If gene A1 has the sequence "AAAAA" in the first strain, but the sequence "AACAA" in the second, then the presence of the SNP at the third allele [3] may indicate that the gene has a role in

[2]Naumann *et al.* define a special form of join operator for this purpose that not only fills in with nulls when values are missing in a source relation, but also merges values when two or more joined views provide a value for a given attribute [15].

[3]Allele is an alternative form of a gene (one member of a pair) that is located at a specific position on a specific chromosome.

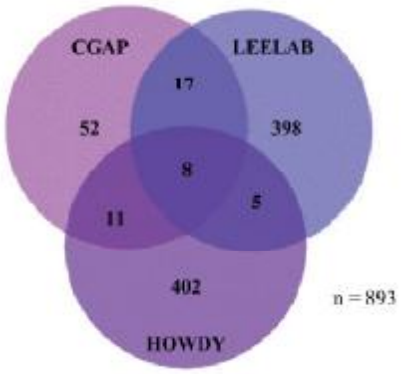


Figure 1: Completeness differences in three human SNP databases involving 74 genes.
(taken from [9])

the development processes that lead to the defect.

SNP data has so far proven to be of value in three main types of genomics analysis, namely, association studies, gene mapping and evolutionary biology studies [6, 19, 21]. In association studies, for example, SNPs have been used to identify genetic factors correlated with complex diseases [5, 7, 24]. Because of this, efforts to discover and document new SNPs have gained momentum in recent years [5, 23], resulting in the establishment of a number of public and private databases [5, 23]. For example, in April 1999, a total of 7000 SNPs had been deposited into the major public databases [5], while by January 2002 some 4 million SNPs had been deposited in the dbSNP database alone [9]. Since then, many other public databases and repositories have been established, including databases like Perlegen [4], GeneSNP [5], PharmGKB [6] and HOWDY [7] [9].

The vast amount of SNP data now available holds out the possibility for supporting many forms of genomic analysis. However, there is a growing concern with the SNP user community regarding the quality of data in these databases, and in particular regarding sharp variations in quality from one database to the next [5, 9, 18]. Although there are some concerns about false positive rates [4], for many scientists, the more immediate concern regards the completeness of the SNP data sets chosen as the foundation for their analyses. For example, Marsh *et al.* undertook a study of three well-known human SNP databases for 74 human genes: CGAP-GAI [8], LEELAB [9] and HOWDY [9]. Their work revealed a significant lack of overlap between the databases, as shown by the Venn diagram in Figure 1. As a result, they cautioned against performing analyses over only a single database (or a small selection) because this would result in incomplete sets of genetic variants being uncovered.

Given this diversity, and the lack of a single reliable reference source for SNPs, a coverage measure such as that

[4]Perlegen - http://www.perlegen.com
[5]GeneSNP - http://www.genome.utah.edu/genesnps/
[6]PharmGKB - http://pharmgkb.org
[7]HOWDY - http://howdy.jst.go.jp/HOWDY
[8]CGAP-GAI - http://gai.nci.nih.gov/cgap-gai/
[9]LEELAB - http://www.bioinformatics.ucla.edu/snp/

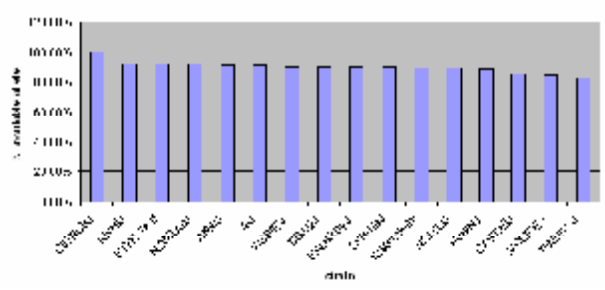


Figure 2: SNP coverage in the Perlegen data set.
(Taken from [12])

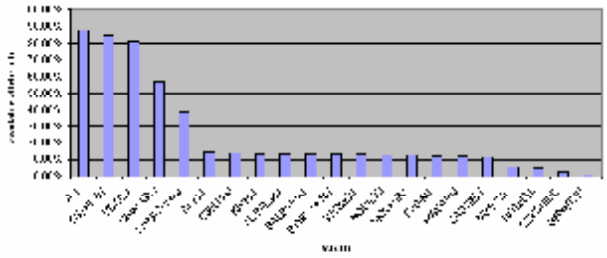


Figure 3: SNP coverage in Ensembl data set.
(Taken from [12])

proposed by Naumann *et al.* [15] would seem to be appropriate. SNP databases tend to contain only SNP records (i.e., they contain horizontal fragmentations of the complete set of SNPs) plus associated metadata, so the universal relation formed by their integration would correspond roughly to the total number of known SNPs. However, completeness relative to the total number of known SNPs is rarely useful in SNP analyses, which tend to be focussed on specific species and specific biological processes, and which are thus concerned with completeness of their data relative to a subset of the available SNP data, rather than the global SNP universe. For instance, SNP studies are typically concerned with a specific set of strains of a specific species, and with a particular region of a particular chromosome thought to be the location of the genes of relevance. An individual scientist, therefore, will be concerned over time not with one standard form of completeness, but with a whole variety of completeness forms, each tailored to the needs of the specific analysis in hand.

To take just one example, Petkov *et al.* undertook an association mapping study, comparing quantitative trait loci (QTLs) for all but the most closely related mouse strains [16]. A QTL is a region of a particular genome that is statistically associated with a specific phenotypic trait. By comparing the SNPs observed in the various strains across the QTLs, the scientists were able to construct a family tree of mouse strains. Their analysis relied on the SNP data sets used having a complete set not of all SNPs or even of all mouse SNPs, but of all SNPs observed in the mouse strains included in the study, and for the QTLs selected for exam-

ination. Even if a more general form of SNP completeness measure had been available to Petko and the team, it would have been of little value unless it could have assessed the specific completeness of data sets relative to these particular criteria (strain, species and QTL). As it was, without any way to assess the validity of their specific assumptions about the completeness of the data sets used in the study, the confidence in the results must be reduced [10].

A more recent study of SNP database completeness discovered that sources can vary widely in the forms of coverage they achieve, as well as in degree of overall coverage. Figures 2 and 3 show the coverage of SNP data in two well known sources for Mouse SNPs: Perlegen and Ensembl [10]. The Perlegen data set resulted from a systematic effort to provide complete coverage across the genome for fifteen selected mouse strains; hence it has good positional coverage, but very poor coverage across the full set of strains (even if only mouse strains are considered). Ensembl, by contrast, is a general repository for SNPs of all kinds (as well as much other biological data). As the figures show, it contains SNPs for a much wider range of strains than Perlegen, but with a much patchier coverage across the mouse genome.

3.2 The SNP Completeness Family

The principal lesson to be drawn from this examination of completeness requirements in an e-Science domain is that even within a single application area we can expect to see not one but many forms of completeness, whether density types measures or (as in this particular case) coverage-style measures. We can also see that certain attributes of the data sets under study make sense as the basis of completeness measures, while others do not. In order to discover the characteristics of an attribute that rule it into one category or the other, it is instructive to attempt to classify the information commonly stored about SNPs according to their potential role in a completeness measure.

Some common SNP attributes are:

1. the unique identifier for the SNP,
2. the chromosome on which the SNP is located,
3. the base position on the chromosome at which the SNP is located (locus),
4. the allele determined for the SNP,
5. the submitter institution or lab which was responsible for experimentally determining the SNP,
6. the strains in which the SNP is known to occur,
7. the species of organism from which the evidence for the SNP was experimentally collected,
8. the "build" (i.e., version) of the species' genome that was used to identify the SNP and in terms of which its position is specified, and
9. the sources where SNPs have been deposited.

Some of these are good candidates for completeness measures. We have already talked about completeness relative to a set of strains of interest (or the full set of strains for an organism), and about completeness of SNP coverage across a chromosome or genome. It might also be reasonable in certain circumstances to ask for a SNP data set that includes all the data submitted by a particular lab (for example, if the lab is known to have undertaken a thorough study of certain QTLs relevant to a particular disease). It seems less sensible to ask for a data set that is complete relative to all possible alleles. There are only a small number of nucleotide bases that occur in DNA (an example, incidentally, of a universe that is very commonly enforced virtually, as a set of letters) and it does not seem likely that a data set which contains an example of a SNP for each of them would have any useful biological or statistical properties. Obviously, the unique identifier for a SNP is going to make a very uninteresting set of SNPs by itself.

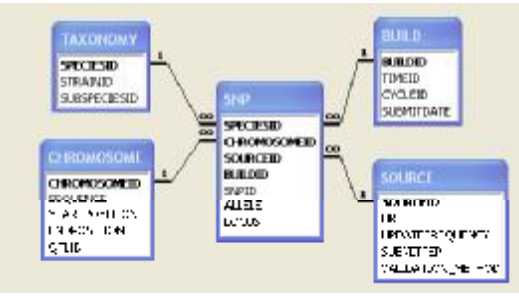


Figure 4: SNP Schema.

If we construct a model of SNP data based around these attributes, it would look something like the schema shown in Figure 4. Readers may notice the similarity of this schema to the star schema form commonly used in data warehouses. If we look further at this model, we see that the attributes around which completeness measures can be envisaged appear in the model as dimensions, while the attributes which are not useful in this way appear on the central fact table. In other words, the completeness attributes refer to a set of values that exists in some sense independently of the SNP data, but which describes its context or meaning (as a kind of metadata). For example, the set of strains is defined independently of the SNP data, but selected strains are associated with specific SNPs in order to describe the context in which the SNP was observed.

What we observe, therefore, is that SNP data is multi-dimensional (including some hierarchical dimensions: a SNP at a particular position is a member of one or more QTLs, which are in turn components of chromosomes, which themselves combine to form a species' genotype). Each such dimension is associated with its own universe. However, the dimension is not measured for completeness by itself. Instead, we measure coverage of the dimension within the data set being measured, a data set in which the main data type is a set of data items. The dimensions fix different applications (e.g., different analysis types), different completeness dimensions will be important to differing degrees and in different combinations. Therefore, the dimensional model actually specifies a whole family of possible completeness measures, which the user should be able to select from as each new analysis type is encountered.

Although we have so far observed this form of multi-dimensional completeness pattern in only this one domain, it seems plausible that it will be of value in others as well,

[10]Ensembl - http://www.ensembl.org/index.html

provided a dimensional model can be created. For example, if we consider a database of clinical patient observations, we might wish to assess its completeness relative to the time at which the observations took place (when studying effects on patients of a time when mRSA was known to be present in certain institutions), to the hospitals at which the observations were made (when wishing to distinguish good and bad practice at institutions), to the demographics of the patients who are the subject of the observations (when attempting to study the course of a disease amongst certain income/occupation groups). Similarly, sales data might be assessed for completeness relative to spatial criteria based on the location of the sale or the type of goods sold.

On the assumption that these multi-dimensional completeness families are of wider applicability than just SNP data, a number of questions arise:

- How can such families of completeness measures be specified?
- How can a user of the family specify that a particular measure (i.e. a member of the family) should be "chosen" for a specific situation?
- What software infrastructure is needed to support the efficient measurement of completeness relative to any of the specific measures that belong to a family?

In the next section, we describe the results of our initial investigations into these questions.

4. MULTI-DIMENSIONAL COMPLETENESS FAMILIES

As we have said, to define a completeness measure for data sets DS, it is necessary to define both the universe of values U and a means of assessing the size of both U and DS. Since our completeness measure families are multi-dimensional, we must also define a set of dimensions that provide the skeleton for the family, and some means by which the completeness scores for the individual dimensions can be aggregated together to give a final value for the completeness of the data set as a whole. We will discuss each of these components in turn.

4.1 Dimension Specification

The most essential part of the specification of a completeness family is a description of the type of data that will be assessed for completeness (the "fact" data) and the dimensions along which its completeness will be measured. The UML class diagram shown in Figure 5 outlines the information to be provided (along with some further components which will be described later in this section). We assume that there is just one "fact" data type, but that it may have multiple attributes. There may also be one or more dimension attributes, which may be grouped together into hierarchies.

As well as defining the individual measures within a family, the specification of the measure includes a set of rules and a schema that will be used to represent the data values during completeness assessment.

4.2 Universe Specification

For each dimension, we need information about the universe of values that represent its complete extent. It may

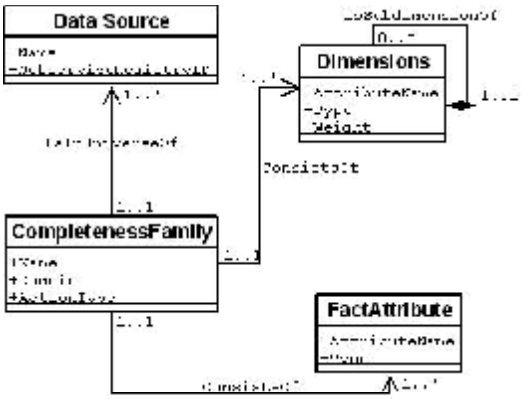


Figure 5: A Completeness Family Model.

be possible in some cases that the universe for a dimension may be specified correctly either by intensional or extensional means. In general, however, we cannot rely on either of these possibilities. We therefore adapt the idea of universal relations described earlier to our multi-dimensional setting. As with the proposal from Naumann *et al.* [15], we assume that the person specifying the completeness family is able to identify a collection of data sources that will together make up the universe for all dimensions for the domain. An individual universe then consists of the union of all discrete values that appear in the dimensional attribute in all the data sources defined by the family designer.

For example, consider the (unreasonably small) SNP data sources shown in Tables 1 and 2. If these two sources are specified as forming the basis of the universe for a SNP completeness family, then the universe for the *species* dimension will be:

$$\{ \textit{"Mus musculus"} \}$$

and the universe for the *strain* domain will be:

$$\{ \textit{"A/J"}, \textit{"129S1/SvImJ"}, \textit{"BTBR T+ tf/J"}, \textit{"DBA/2J"}, \textit{"CAST/EiJ"}, \textit{"C3H/HeJ"} \}$$

Since the format of the data sources specified as the universe for the family may vary dramatically, it is also necessary for the family designer to provide for each data source the URL of a Web service that will extract the sets of discrete values contained within it for each dimension attribute. The data should be provided in the format described by the dimensions and the fact table.

4.3 Dimensional Completeness

We adopt a simple ratio metric for the completeness measures for individual dimensions. If the data set being assessed contains the set v_i of discrete values for dimension i, then its completeness relative to the specified universe for that dimension, u_i, is given by:

$$c_i(ds) = |u_i \cap v_i| / |u_i|$$

This is a standard equation that is built into the family model. There is no further information for the designer to specify at this stage.

SNPID	SPECIES	STRAIN
rs2020841	Mus musculus	A/J
rs2030843	Mus musculus	A/J
rs2040845	Mus musculus	A/J
rs2060840	Mus musculus	129S1/SvImJ
rs2070849	Mus musculus	BTBR T+ tf/J

Table 1: SNP Data Source 1.

SNPID	SPECIES	STRAIN
rs1020841	Mus musculus	A/J
rs1030843	Mus musculus	DBA/2J
rs3040845	Mus musculus	CAST/EiJ
rs6070849	Mus musculus	BTBR T+ tf/J
rs4070849	Mus musculus	C3H/HeJ
rs5070849	Mus musculus	C3H/HeJ

Table 2: SNP Data Source 2.

4.4 Aggregating Dimensional Completeness

Since users of a family may be concerned with more than one completeness dimension at a time, we must have some way of aggregating the individual dimension scores just described into a single over-arching completeness score for the data set as a whole (relative to the dimensions of interest). In this first version of the family model, we adopt a simple weighted average approach, since this will preserve the ratio nature of the score as well as balancing the strengths and weaknesses of the selected dimensions against one another. Other approaches may well prove to be more appropriate after further work.

Rather than fixing the weightings, we allow designers of families to specify their preferred default weights for each dimension. As we shall later discuss, these weights can be overridden at runtime, when a specific completeness measure from within the family is invoked.

4.5 Completeness Families in Use

Given the information just described, our goal is to (automatically, as far as possible) generate a software component or components that can implement the completeness family. We use as the basis for this generation the framework provided by the Qurator project[11]. The Qurator framework supports model-driven generation of components for IQ assessment called *quality views* [11]. A quality view or QV for short (illustrated in Figure 6) is a layered component that takes in a data set, classifies each element in the data set according to its quality according to some domain specific measure, and outputs a version of the data set, transformed in some way according to the classification. For example, a very common transformation involves filtering out elements of the data set that do not meet some predefined quality standard. The QV also takes in some arbitrary parameters, and records a report giving the quality classifications assigned to each item of the input data set.

As the figure shows, internally, quality views are layered components. The first layer consists of components that gather evidence about the data elements that pertains to their quality. The middle layer takes this evidence as input, and applies a decision procedure for classifying each data set element in terms of its quality, as indicated by the evidence

[11]www.qurator.org

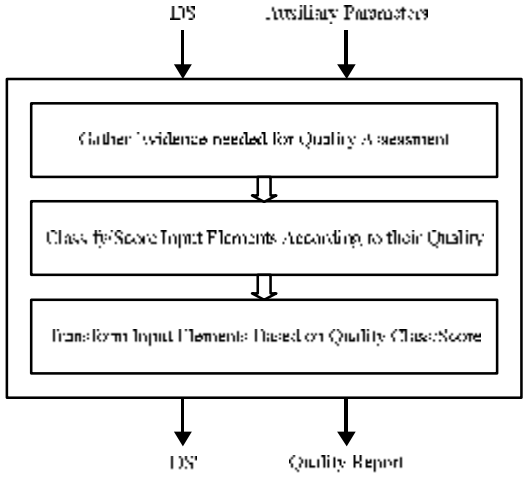


Figure 6: The Qurator Quality View Pattern.

required. These quality classifications are then passed on to the third layer, the "action" layer, which transforms the input elements based on their quality.

QVs can be specified as a high level model which is then compiled into a Web service that implements the black-box behaviour just described. These Web services can then be incorporated into the user's preferred information manipulation environments, to make them "quality aware". Ideally, we would like our completeness families to be manifested as QVs, so that they are easy to adopt for users already familiar with other forms of IQ measurement in QV form. Since we can pass arbitrary parameters to QVs, this is relatively easy to manage.

Completeness is an aggregate measure that can only be sensibly applied to sets of data, rather than to individual values. Therefore, the input data set for a completeness family QV (cfQV) must be a set of data sets, each of which is to be assessed for completeness. (Of course, the set can be a singleton set if only one data set is to be assessed for completeness.) In this first version of the cfQV model, we assume that the caller wishes to assess the completeness of one or more of the sources specified in the universe for the family, and therefore the input data set is simply a list of the identifying names given to the sources of interest when the family was defined. In future version of our work, we will expand on this simplistic definition.

The caller must also specify which dimensions are to be used for assessment of completeness, i.e., which of the collection of completeness measures embodied by the family are to be invoked on this occasion. The dimensions are specified as a list of names in an arbitrary parameter to the QV. They can optionally be accompanied by weights for the dimensions if the caller does not wish to make use of the default weights defined in the family.

How, then, are the three layers of the QV structure employed during the execution of a specific completeness measure? Since the task of the first layer, the evidence gathering components, is to collect objective evidence about the quality of the inputs, in a cfQV this layer has the task of gathering the completeness scores for the individual dimen-

sions. The exact nature of this computation is described in Section 5. The middle layer applies the weighted average to the individual dimension completeness scores, and produces an overall completeness score. The action layer then performs whatever action has been specified by the designer of the family (see Figure 5), for example, as an XSLT rule set.

The model shown in Figure 5 contains all the information needed to allow us to generate this form of QV automatically. However, a further important design choice has yet to be made, as we shall discuss in the next section.

5. SOFTWARE INFRASTRUCTURE

The simplest implementation of the completeness family quality views just described is one in which all the information needed for completeness assessment is gathered from the universe of sources on demand, at the time when the cfQV is invoked for some specific measure. However, this is unlikely to be very efficient, especially if the number of sources in the universe is large, if the universes for any of the dimensions are large, or if the number of dimensions of interest to users is significantly less than the total number available. If the cfQV is to be invoked regularly, then an alternative approach would be to materialise the various dimensional universes in central warehouse. Although the effort of constructing and periodically maintaining this warehouse might be considerable, it could be worthwhile if a large number of users are accessing the family, each wishing to make use of a slightly different combination of criteria and/or weights.

This would require some infrastructure beyond the borders of the QV component, a departure from our earlier work on QVs. We therefore chose to explore this option, with the aim of determining the costs and potential benefits of the approach, and of discovering how much of the necessary software infrastructure could be automatically generated from the completeness family model. Figure 7 shows the full set of components. At the top left, we have the specification of the completeness family, in terms of the high-level model. This is used to create and configure the components needed for the rest of the infrastructure. On the right of the diagram is the infrastructure needed to create the warehouse of universe data needed to support the cfQV. We have already noted the similarity of our data representation (shown in Figure 4) to a star schema. Because of this, we chose to use data warehouse technology to implement the materialised universe view [12]. This supports queries over a wide variety of combinations of dimensions and sources, so that the full range of completeness measures embodied by the family can be efficiently and seamlessly supported.

The warehouse is populated from the sources using the Clover ETL framework[13]. Data is extracted from the sources using the specified web services, and is then transformed and loaded into the materialised universe source. At present, we have considered only the initial creation of this warehouse. In future, we will of course have to consider the infrastructure needed to maintain it when the underlying sources change. However, in essence, this is a very standard warehouse refresh process, and we should be able to make use of the wealth of expertise and tools available in this area.

[12]MySQL (http://www.mysql.com) is used as the database management system.

[13]Clover.ETL - http://cloveretl.berlios.de/

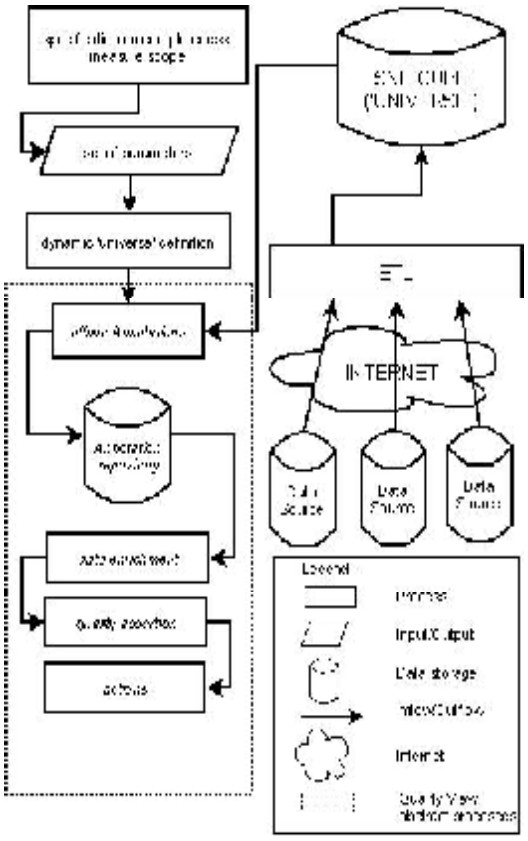


Figure 7: The cfQV Software Infrastructure.

At the bottom left of the figure is the QV component that is created automatically from the cfQV model in order to act as the point of access to individual members of the completeness measure family. When this QV is invoked, the evidence gathering functions issue queries to the materialised universe in order to compute the completeness of the selected sources relative to the selected dimensions. For example, suppose that a cfQV has been created for SNP data, with a variety of well known SNP sources as universes, and the dimensions shown in Figure 4. Suppose further that some user invokes the resulting cfQV requesting that completeness of the Perlegen and Ensembl data sources relative to the dimensions of *strain* and *chromosome*, with equal weights. The quality evidence function would begin by issuing the following query to warehouse that stores the family universes:

```
SELECT DISTINCT(t.strainid),
       DISTINCT(s.chromosomeid)
FROM taxonomy t JOIN snp s
ON t.speciesid = s.speciesid;
```

It then retrieves the set of values in the specific sources, it issues queries of the form:

```
SELECT DISTINCT(t.strainid),
       DISTINCT(s.chromosomeid)
```

```
FROM taxonomy t JOIN snp s
ON t.speciesid = s.speciesid;
WHERE s.sourceid = "perlegen";
```

The resulting ratios are passed on to the quality classification layer of the cfQV, which computes their average, and outputs the results in the quality report.

Table 3 shows an example of the SNP cfQV at work.

6. CONCLUSION

The work presented in this paper stems from the observation that, in several applications domains, the notion of data completeness can be expressed quite naturally in terms of multiple dimensions. In the context of SNP data analysis for biological applications, for example, we have been able to identify a number of dimensions (including chromosome, strain, and species) that scientists can use, either individually or in combination, to express useful measures of completeness. Our definition of dimensional completeness assumes that, for each dimension, completeness is measured with respect to a universe of values that is available for that dimension. This leads naturally to the idea of expressing complex completeness measures as OLAP-type queries on a dimensional schema.

The study presented in the paper is a preliminary investigation into this idea, and is limited to a few, simple example queries. Nevertheless, we have shown how dimensional completeness measures can be supported in practice by leveraging our existing Qurator framework [11], proposed in earlier work. In particular, we are implementing a first prototype to show how we can rank a collection of data sources (e.g., a collection of SNP databases) by extending the Quality View model that is at the core of Qurator. In the full implementation, users will be able to specify the ranking criteria at run-time, as aggregation queries on the dimensional completeness model.

7. REFERENCES

[1] D. P. Ballou and H. L. Pazer. Modeling data and process quality in multi-input, multi-output information systems. *Management Science*, 31(2):150–162, 1985.

[2] D. P. Ballou and H. L. Pazer. Modeling completeness versus consistency tradeoffs in information decision contexts. *IEEE Transactions On Knowledge and Data Engineering*, 15:240–243, 2003.

[3] C. Batini and M. Scannapieco. *Data Quality : concepts, methodologies and techniques*. Springer, Berlin, 1998.

[4] A. Brass. private communication, 2007.

[5] A. J. Brookes. The essence of SNPs. *Gene*, 234:177–186, 1999.

[6] L. Comai and S. Henikoff. TILLING: practical single-nucleotide mutation discovery. *The Plant Journal*, 45:684–694, 2006.

[7] E. Halperin, G. Kimmel, and R. Shamir. Tag SNP selection in genotype data for maximizing SNP prediction accuracy. *Bioinformatics*, 21:195–203, 2005.

[8] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation : Practice and Experience*, 18:1039–1065, 2005.

[9] S. Marsh, P. Kwok, and L. H. Mcleod. SNP database and pharmacogenetics: Great start, but a long way to go. *Human Mutation*, 20:174–179, 2002.

[10] A. Martinez and J. Hammer. Making quality count in biological data sources. In *IQIS*, pages 16–27. ACM, 2005.

[11] P. Missier, S. Embury, R. Greenwood, A. Preece, and B. Jin. Quality views: capturing and exploiting the user perspective on data quality. In *Proceedings of the 32nd international conference on VLDB '06*, pages 977–988. ACM Press, 2006.

[12] P. Missier, S. Embury, C. Hedeler, M. Greenwood, J. Pennock, and A. Brass. Accelerating disease gene identification through integrated SNP data analysis. In *Proceedings 4th International Workshop on Data Integration in the Life Sciences*, pages 215–230. Springer, 2007.

[13] A. Motro. Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480–502, 1989.

[14] A. Motro and I. Rakov. Estimating the quality of databases. In *FQAS '98: Proceedings of the Third International Conference on Flexible Query Answering Systems*, pages 298–307, London, UK, 1998. Springer-Verlag.

[15] F. Naumann, J. Freytag, and U. Leser. Completeness of integrated information sources. *Information Systems*, 29(7):583–615, 2004.

[16] P. M. Petkov, Y. Ding, M. A. Cassell, W. Zhang, G. Wagner, E. E. Sargent, S. Asquith, V. Crew, K. A.Johnson, phil Robinson, V. E. Scott, and M. V. Wiles. An efficient SNP system for mouse genome scanning and elucidating strain relationship. *Genome Research*, 14:1806–1811, 2004.

[17] T. Redman. *Data Quality for the Information Age*. Artech House, Boston, MA, 1996.

[18] D. E. Reich, S. B. Gabriel, and D. Atshuler. Quality and completeness of SNP databases. *Nature Genetics*, 33(Brief Communication):457–458, 2003.

[19] D. Savage, J. Batley, T. Erwin, E. Logan, C. G. Love, G. A. Lim, E. Mongin, G. Barker, G. C. Spangenberg, and D. Edwards. SNPServer: a real-time SNP discovery tool. *Nucleic Acids Research*, 33(Web Server Issue):493–495, 2005.

[20] M. Scannapieco and C. Batini. Completeness in the relational model: a comprehensive framework. In *MIT Conference on Information Quality (IQ)*, pages 333–345, 2004.

[21] S. Sherry, M. Ward, J. Baker, E. Phan, E. Smigielski, and K. Sirotkin. dbSNP: the ncbi database of genetic variation. *Nucleic Acids Research*, 29:308–311, 2001.

[22] R. Wang and D. Strong. Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems*, 12(4):5–33, 1996.

[23] H. Xu, S. G. Gregory, E. R. Hauser, J. E. Stenger, M. A. P. Vance, J. M. Vance, S. Zuchner, and M. A. Hause. SNPselector: a web tool from selecting SNPs for genetic association studies. *Bioinformatics*, 21(22):4181–4186, 2005.

[24] S. Ye, S. Dhillon, X. Ke, A. R.Collins, and I. N. Day. An efficient procedure to genotype haplotypes and single nucleotide polymorphisms. *Nucleic Acids Research*, 29(17):1–8,

DATABASE	STRAIN SET	RELATIVE STRAIN RATIO,weight=1	CHROMOSOME SET	RELATIVE CHROMOSOME RATIO,weight=1	OVERALL COMPLETENESS (Strain Ratio + Chromosome Ratio)
PERLEGEN	5	(5/30)(1)=0.17	5	(5/22)(1)=0.23	0.40
ENSEMBLE	20	(20/30)(1)=0.67	15	(15/22)(1)=0.68	1.35
UNIVERSE	30	(30/30)(1)=1.00	22	(22/22)(1)=1.00	2.00

Table 3: An Example of Annotated SNP Completeness Evidence.

2001.

Managing Data Quality in Business Intelligence Applications

Florian Daniel, Fabio Casati, Themis Palpanas, Oleksiy Chayka

University of Trento
Via Sommarive 14
38100 Povo (TN) - Italy

{daniel, casati, themis, chayka}@disi.unitn.it

ABSTRACT

Business Intelligence (BI) solutions commonly aim at assisting decision-making processes by providing a comprehensive view over a company's core business data and suitable abstractions thereof. Decision-making based on BI solutions therefore builds on the assumption that providing users with targeted, problem-specific fact data enables them to make informed and, hence, better decisions in their everyday businesses. In order to really provide users with all the necessary details to make informed decisions, we however believe that – in addition to conventional reports – it is essential to also provide users with information about the quality, i.e. with quality metadata, regarding the data from which reports are generated. Identifying a lack of support for quality metadata management in conventional BI solutions, in this paper we propose the idea of quality-aware reports and a possible architecture for quality-aware BI, able to involve the users themselves into the quality metadata management process, by explicitly soliciting and exploiting user feedback.

1. INTRODUCTION

Over the last years we have been witnessing an increasing use of *Business Intelligence* (BI) solutions, i.e., solutions such as data warehouses, reporting and data mining tools that allow business people to query, understand, and analyze their business data in order to make better decisions. As it is well known, the quality of the BI solutions is at most as good as the quality of the data in input. Bad or low-quality data may lead to bad business decisions. Imagine, for example, that the Department of Health wants to predict the quantity of flu drugs that is expected to be used in winter 2008/2009, to prepare for outbreaks or simply to negotiate discount rates with drug manufacturers. If the prediction is based on low quality data, e.g., data that are old, incomplete or incorrect, an insufficient quantity of drugs may be predicted and negotiated. Also, while purchasing additional quantities of drugs at higher prices might be acceptable, there still remains the danger that additional drugs cannot be delivered timely, as the manufacturer might not be able to quickly respond to late orders. Analogous problems may occur when logistics departments take goods-routing and warehousing decisions based on wrong sales or shipment data.

VLDB '08, August 24-30, 2008, Auckland, New Zealand.

Data quality problems in data warehousing and BI applications are more and more common (and more and more impacting the everyday business) due to the fact that warehouses are becoming tentacular, reaching to a larger and larger number of source systems, also due to the recent trend towards enterprise-wide data warehouses. Different source systems typically provide data at different levels of quality, and the ETL process also becomes complex with the risk of errors in the cleaning procedures.

The above scenario underlines two different kinds of problems, whose combined effect leads to wrong business decisions. First, the low quality of the data. Second, the lack of awareness by the analysts that the data is of low quality and therefore that the reports they see and based on which they take their decisions are, in fact, inaccurate.

The latter problem and an attempt towards its resolution or mitigation is the focus of this paper. In particular, we propose the notion of *quality-aware reports* in BI applications, where reports explicitly expose the quality of the data underlying the generated results and, most importantly, their effect on the quality of the report. If the Department of Health were aware of the low quality of the data in input, it could for instance have done some further investigation to refine the quality of data in input and the prediction, thus saving money and assuring on-time delivery.

From an IT perspective, the above problem implies the ability to i) associate quality metadata with a report, ii) compute this metadata based on quality information on the base data, and iii) display such information to users in an easily comprehensible and "actionable" way, so that the viewers can identify the quality problems, understand their extent, and decide how relevant/severe they are and what to do about those. An interesting challenge here is represented by the fact that there are many different potential quality problems (from late arrival of the data, to potentially incorrect information at the source, to inconsistent use of terms by the persons doing data entry, to entity duplication issues, and many more), and it is important that users are aware of *why* a report is considered of low quality and which parts of the report have problems. For example, a report on the total number of surgeries may be inaccurate because it lacks data for the month of May from St. John's Hospital. The report analyst may decide that this is a serious issue as St. John's is large and has a highly variable number of procedures, or may decide that it is irrelevant as the number of procedures is very low or it is fairly stable month over month (or, very pragmatically, the analyst may make a call to St.

John's and ask an estimate for the data value). Hence, it is not sufficient to merely show quality information; we need to make the user aware of the provenance of the quality information and of the reasons for certain assumptions on the quality of a report.

The latter observation also underlines that quality is *subjective* in two ways: first, the quality issues may or may not be significant or impacting a certain decision. Second, analysts may have (and, in our experience, very often do) personal knowledge or opinions on the quality of the data. For example, a health analyst may know or anyway believe that data from St. John's is often inaccurate, or that doctors use terms inconsistently when they enter data, or that the data collection process is entirely manual and therefore subject to frequent errors. From an IT perspective and with respect to the goal of building a quality-aware report solution, this observation has two implications.

First, we need to allow users to define personalized *quality-aware views* on reports or in general on the data (in contrast to *quality views* introduced in [22] and [23], where quality views express a users' quality processing requirements in terms of workflows). These quality profiles would embody any knowledge the user may want to express over the data. Such knowledge may not always be structural (as in the example of the lack of confidence in St. John's data), but also *situational.* For example, the user may see a detailed report on surgical procedures and detect that two different entries correspond in fact to the same procedure and therefore should be merged. For the situational case, this means that the definition of the quality profile can involve report-specific information, and can be interactive, that is the user "plays" with the report quality to correct the information when needed or to have the quality metadata take into account the user's personal beliefs on the data quality. Allowing the user now to interactively include/exclude data or to merge/unmerge records and to recompute reports on the fly would allow him/her to understand the importance of including/excluding such data into/from the report and to act accordingly.

Second, the opportunity arises for capturing user feedback and personalized quality-aware views and for using this information for re-defining the way quality metadata is computed. For example, if several users note that St. John's data is not to be trusted, then this information may be considered to be accurate by the BI applications (perhaps after review by an authorized user) and used to refine the quality metadata for reports that use St John's data.

As a final observation, we point out that the problem of data quality awareness in BI applications is not restricted to reports, but also applies to data mining models that mine data to discover information. The challenge here is how mining models can take into account data quality when computing their results, and how mining algorithms should be modified in this respect.

In this paper we present an architecture for quality-aware BI solutions and discuss some of the fundamental issues behind it, and specifically i) which are the main ingredients of such a solution and the related challenges; ii) which are the quality dimensions relevant in BI and how to model quality metadata for reports; iii) how quality in the base (warehouse) data affects quality in reports, and hence how to map base quality metadata into report quality metadata; iv) how to model quality-aware views (also called quality profiles); v) how to structure user interaction with the reports.

As the problem space is huge, and as the research is still in the early stages – consistently with the spirit of a workshop, this paper presents issues and solutions we experienced, rather than complete solutions – we focus on some of these issues, specifically quality for BI applications and the relation between data quality and report quality.

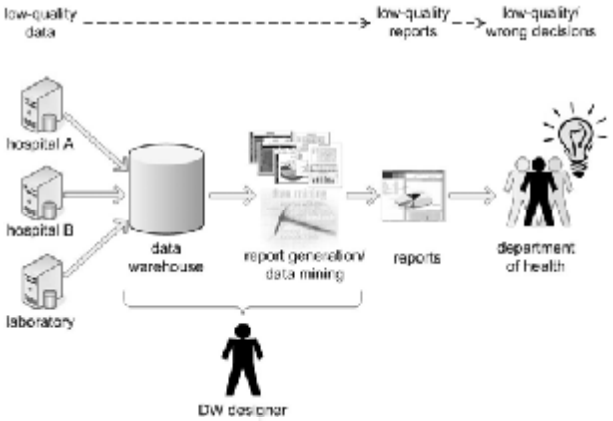


Figure 1: The risk of low-quality data in healthcare BI.

2. TOWARDS QUALITY-AWARE BI

2.1 Reference Scenario

Throughout this paper we will be using the healthcare example as our reference scenario, in order to exemplify and better explain our ideas. Specifically, we are interested in assisting the Italian Department of Health in forecasting the quantity of flu pharmaceuticals (e.g., aspirin) for the upcoming winter 2008/2009, in order to enable the Department of Health to agree with manufacturers on nation-wide stable and fare prices for its citizens.

The typical levers in the hands of the Department of Health to control pharmaceutical prices are dedicated tax regulations (e.g., lowering the value added tax for individual pharmaceuticals) or participation in the production cost (e.g., the state may take over part of the manufacturing cost of a pharmaceutical, in order to keep its customer price low). Obviously, each intervention by the Department of Health is associated with a cost for the State: either there is a missing income in terms of taxes that are not levied, or there is an expense in integrating manufacturing costs. Either way, the expected cost for the state needs to be predicted after summer, when the Government prepares the budget for the following year.

For the prediction of the pharmaceutical demands, the Department of Health adopts a BI solution that sources data from each of the country's 20 Regions (Italy is politically and geographically structured into Regions, at a higher level, and Provinces at a lower level), which aggregate the necessary healthcare data from their local hospitals, laboratories, emergency rooms, and the like. Regional data are collected in a centralized data warehouse, which enables the Department of Health to view National health reports, to analyze and mine collected data, and to predict pharmaceutical demands. Figure 1depicts the described scenario.

Figure 1 also highlights the core problem the Department of Health has to deal with: low-quality data in input unavoidably lead to low-quality reports in output. Low-quality reports might lead to wrong estimates and unwanted budget problems. Unfortunately, low-quality data is a reality, and it is hard (if not infeasible) to eliminate all possible quality problems via data cleaning during the ETL process. For instance, Figure 2 exemplifies some

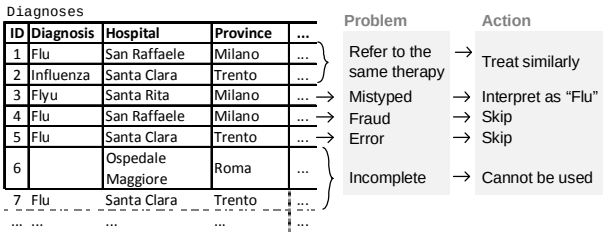


Figure 2: Typical data quality problems. Dashed lines represent expected but missing data.

typical quality problems we might encounter when looking for example at the *Diagnoses* table containing information about the diagnoses made in different hospitals.

The first two rows refer to the same diagnosis, with the only difference that in the first row the diagnosis is "Flu", in the second row the diagnosis is "Influenza"; it is important to understand that both rows actually refer to the same diagnosis. The diagnosis in row number 3 is mistyped, which makes it algorithmically hard to understand that row 3 refers to flu, as well. The previous cases represent inconsistencies in the data, which might lead to too low a prediction of necessary flu pharmaceutical, if not identified as such. But even if we are able to infer that the three rows might refer to the same diagnosis, we typically will not be entirely sure of this finding, and it might be good to keep track of our level of confidence when further processing the aligned data. In row number 4 we have assumed that it simply does not correspond to the truth, in the sense that a doctor could simply have declared a fake diagnosis in order to get money for a not provided treatment; the row should actually not be considered in the computation of reports. This kind of fraud is very hard to identify in practice and, hence, might lead to an incorrect overestimation of the drug demand. Row number 5 represents another possible problem: a simple error. Errors happen, but we should be able to identify them, in order to skip the respective row. An erroneous tuple such as the one in the figure might for example be due to a test of the source system where test tuples have not been eliminated correctly. Identifying such kinds of errors is however practically impossible and they might lead to an overestimation in the prediction. Finally, row number 6 lacks the value for the diagnosis attribute, and row number 7 is simply missing (represented by the dotted borders); hence, those rows cannot be used, even though they might correspond to a flu diagnosis (but we don't know). Incomplete data might lead to an underestimation of the drug quantity.

Although the above examples highlight only few of the typical quality problems in databases, they are still enough to show how low-quality data in input to the BI solution might negatively affect the quality of the output of the BI solution, i.e., of the reports and the data mining results.

2.2 Research Challenges and Contributions

Generalizing the described healthcare scenario allows us to identify the research challenges that characterize the data quality problem of most business intelligence scenarios:

- How to *define quality* and identify measurable *quality properties* that appropriately characterize the specific case of data warehousing and data mining.
- How to model *quality metadata* associated with warehouse data and reports.
- How to *map* warehouse quality metadata into report quality metadata (given a report definition as a query over warehouse data).
- How to *improve* the quality of data in the warehouse, e.g., via data cleaning techniques.
- How to effectively *expose quality metadata* and assumptions (e.g. about ETL procedures or data cleaning decisions) to end users. Interactive, quality-aware reports could for instance allow dynamic what-if scenarios based on data quality properties.
- How to collect and manage *quality-related user feedback*. While the automated data cleaning process might help mitigate quality problems, in many cases the best evaluator of quality is still the user. Effectively collecting explicitly-provided user feedback might for instance help fine-tune the data cleaning process and improve the quality of outputs.
- How to compute *customized reports* based on individual user feedback. A user might provide customization instructions for his/her reports, expressing personal preferences or knowledge about the quality of data underlying the reports.
- How to *propagate collective user feedbacks* into the warehouse and ETL procedures. If a specific quality problem reaches a predefined threshold of aligned user feedbacks, the feedback might be transformed into proper quality metadata to be used globally in the warehouse.
- How to *build quality-aware mining models* from quality-labeled data. It might for instance be interesting to reconsider known mining models, however considering quality in the training and validation datasets.

As a first step toward quality-aware BI and in particular focusing on the problem of understanding how to inform and involve the end user in report quality management, in this paper we provide the following contributions:

- We define *quality in the context of data warehousing* by identifying relevant quality properties.
- We define the *concept of quality-aware report* as a means to provide users with an awareness of the quality of the data underlying the reports they are inspecting.
- We propose a *quality-aware data warehouse architecture* that aims at i) managing quality metadata, ii) enabling the computation of quality-aware reports, and iii) taking into account user-provided feedback.
- We discuss the *modeling* of warehouse and report metadata and the mapping between the two in report computation.
- We discuss *related works* in light of the requirements identified for the development of the envisioned quality-aware data warehouse and position our work accordingly, highlighting still *open research challenges*.

3. DATA QUALITY IN BI

3.1 The Notion of Data Quality

To assess the quality of data, the research community has identified various dimensions. A common set of quality dimensions and their definitions (proposed by [41]) is listed in Table 1.

Table 1: Commonly accepted data quality measures [41].

Dimension	Definition
Accessibility	the extent to which data is available, or easily and quickly retrievable
Appropriate amount of data	the extent to which the volume of data is appropriate for the task at hand
Believability	the extent to which data is regarded as true and credible
Completeness	the extent to which data is not missing and is of sufficient breadth and depth for the task at hand
Concise representation	the extent to which data is compactly represented
Consistent representation	the extent to which data is presented in the same format
Ease of manipulation	the extent to which is easy to manipulate and apply to different tasks
Free of error	the extent to which data is correct and reliable
Interpretability	the extent to which data is in appropriate languages, symbols and units, and the definitions are clear
Objectivity	the extent to which data is unbiased, unprejudiced and impartial
Relevancy	the extent to which data is applicable and relevant for the task at hand
Reputation	the extent to which data is highly regarded in terms of its source or content
Security	the extent to which access to data is restricted appropriately to maintain its security
Timeliness	the extent to which data is sufficiently up-to-date for the task at hand
Understandability	the extent to which data is easily comprehended
Value-added	the extent to which data is beneficial and provides advantages from its use

The dimensions above, and analogous proposals arising from the data quality community, are oriented toward evaluating the quality of a generic dataset. In this paper we focus on the problem of representing data quality to end users in BI applications, with the specific goal and challenge of helping users understand the quality of BI results presented to them and to avoid making wrong assumptions on the data presented and therefore running the risk of making wrong decisions. We are particularly interested in exposing non-obvious quality problems to end users, rather than in quality issues that are presented by design in the BI applications.

For example, we are not interested in discussing timeliness (freshness) of the data in the warehouse, or in trying to understand if a warehouse with data loaded monthly is "good" or "bad". Similarly, we are not interested in assessing completeness of sources in the sense of ensuring that we have deployed our ETL application to extract data from all possible hospitals or social care structures. These are conscious *design* decisions, which are well known to the end user (or which anyway can be easily communicated). Such dimensions as appropriate amount of data, concise representation, ease of manipulation, relevancy, security, understandability and value-added are also irrelevant for our goals.

Instead, we are interested in spotting situations where data is loaded monthly but for a given batch load one source did not make the data available, or the data was not loaded due to ETL errors. Similarly, we are interested in data incompleteness problems caused by a source not logging (or the ETL not extracting) some of the surgical procedure data for certain patients or class of surgical procedures. The above situations may lead users to view aggregated data based on certain assumptions (all surgical procedures data is there) which may turn out to be incorrect in the specific report they are viewing.

In summary, we focus on quality dimensions relevant in multi-source BI applications for the purpose of communicating to the end users transient properties of the data sources and of the data extracted from them and loaded into the warehouse.

3.2 A BI-Specific Definition of Quality

For the purpose of developing quality-aware reports in BI applications, we propose the following quality dimensions as the relevant ones:

- Completeness
- Consistency
- Confidence

Completeness measures to which extent data that according to the warehouse specifications should have been recorded in a table are effectively present. We refer to *vertical incompleteness* when we measure completeness of data in a column, that is, the quantity or percentage of values in the column that are null (or where codes such as “9999” are inserted in place of missing information; for an example, see Figure 3) when the information is instead supposed to be there. Null values might in general be allowed and represent meaningful information, e.g., for persons that undergo their first-ever surgery, the fact that the date of previous surgery is null is acceptable and not a sign of incompleteness. *Horizontal incompleteness* refers instead to the quantity or percentage of entire tuples (e.g., tuples representing surgical procedures), typically entire facts or dimension entries that have not been recorded. Figure 3 exemplifies this dimension, showing for example that row number 8 which should have been logged is not present in the recorded dataset. Note that in [25] vertical and horizontal completeness are called *density* and *coverage*, respectively.

There are many reasons why incompleteness may occur, such as errors or omissions in the data entry at the source, or errors in the ETL process that fails to record some of the tuples in the warehouse. An important and relatively frequent problem that leads to incompleteness is *batch unavailability*, that is, delays in loading batch data in the warehouse. In addition to vertical and horizontal completeness (like [25] and [13]), we therefore also consider batch availability as completeness property. Sources are supposed to make data available at specified time intervals, which is when the data load into the warehouse occurs. A batch is unavailable if the source does not provide the data or if the ETL process fails for some reason (e.g., it is unable to connect to the source). Unlike other incompleteness scenarios, batch unavailability is relatively easy to detect and to communicate to the report viewers. In the table in Figure 3, for instance, the entire batch from the Province of Bolzano is missing.

Completeness can be modeled as *extensional* or *intensional* metadata, and at different levels of granularity. Extensionally, completeness for cells is expressed as a binary value (i.e., true/false). For columns and tuples, it is expressed in percentages (100% representing full completeness), for each column in case of vertical completeness and for the entire table for horizontal ones.

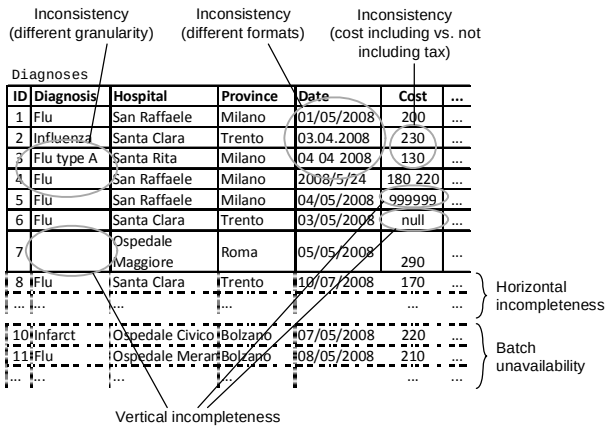


Figure 3: Completeness and consistency problems in the DW. Dashed lines represent incomplete or unavailable data.

Since data warehouses are typically loaded in batches, completeness measures can also refer to batches of load. Indeed, in practice it does happen that different batches may have different degrees of completeness, and, as mentioned, entire batches may be unavailable. Even more frequent is the case where completeness is related to specific data sources. This information is very important not only because we can know, when computing a report, if the report is complete or not (e.g., a report not querying St. John's data may be complete even if St. John's data is incomplete), but also because users can make own judgments on the quality assumption.

The latter two cases and the reasoning above show the need for an intensional measure of incompleteness, where a rule is stored rather than a mere measure (cf. business rules [12], [30]). In general rules are functions over the dataset that identify a set of tuples, and for these tuples define a completeness measure in terms of percentage. A textual description is also attached to the rule. Informally, examples of these rules are "all entries by Dr. Smith are 70% complete on average". Formally, functions can for example be expressed as SQL queries.

Finally, the case of batch (un)availability is measured in terms of which batch loads are (un)available. Each batch is also associated to a time window to which the extraction refers (e.g., batch 22 correspond to May 2008), which is something then useful to provide information at report viewing time. The measure can be associated to a data source (e.g., all data from that source for batch X are unavailable) or to a data source *and* a table (e.g., all surgical procedures data from that source for batch X are unavailable).

In the current paper we do not discuss two important issues: first, how to derive (compute) the intensional or extensional measures, and second, how to deal with conflicting intensional rules. These problems, especially the first, are very hard and can be subject of entire lines of research [25], [33]**.**

Consistency denotes the uniformity of the information in a given column. *Syntactic* consistency refers to uniformity in the data format (e.g. different date formats in the table in Figure 3). This is typically something that is detected and corrected at data cleaning time (e.g. via normalization), and is not discussed further.

Semantic consistency refers to the satisfaction of semantic rules defined over a set of data items. There are different reasons why the information can be inconsistent (see Figure 3):

1. Different understanding of the semantics of the field: For example a date field may refer to the patient surgery date, or to the date the diagnosis was made.
2. Different abstraction/precision level: the semantics of the field may be commonly understood, but the degree of precision or detail when entering the data can be different. For example, a doctor may generically enter "Flu" while another can enter "Flu type A" which is more precise.
3. Different units: in this case the understanding and granularity are the same but the interpretation differs on the unit of measure, such as Celsius vs Fahrenheit or, as depicted in Figure 3, cost including taxes vs. cost excluding taxes.

From an assessment perspective, consistency is often checked by defining a set of business rules [12], [30], and its measure may take various forms: first, there can be a qualitative measure attached to a data set to denote if the values there are overall consistent or not. However inconsistencies typically occur between data sources, or between different persons entering data. For this reasons, (in)consistency is also expressed in terms of:

- A measure of "inconsistency" applied to a data cell whose value is suspected to be inconsistent with the other values (a fine-grained quantitative measure is meaningless here, while a qualitative distinction with a few, possibly as few as two distinct values for (in)consistency suffice for our purposes).
- An intensional description that labels as inconsistent or possibly inconsistent the data from a given source or entered from a data entry agent.

In general the approach to intensional description is the same as for completeness: a description of an inconsistency is represented by a textual description, by a function over the data and the warehouse metadata (provenance and batch information, such as data source or time or data load) that identifies tuples affected by the quality issue, and by the inconsistency problem. As an example again pounding on Dr Smith, we can state that diagnosis data entered by Dr. Smith and related to flu is inconsistent.

Confidence describes the perceived accuracy of the data, or the degree of trust (or, from the opposite perspective, the degree of uncertainty) that the data present in a table or set of tables is accurate. In general, reasons for marking data as uncertain (low confidence) include lack of trust in a data source, potential errors or uncertainty detected during data cleaning [31], [16], outlier values [27], and others. As for the above measures, we can define confidence as a probability (certainty) measure associated to cells, tuples, tables, or data sources, as for example done in Trio [4].

However confidence has more sides that need to be addressed and that cannot be covered by the above representation. A key problem is that a large number of uncertainty issues in data warehouses are caused by *entity resolution* issues. This means that a confidence representation must include the possibility of expressing that two or more tuples may refer to the same real world entity. In addition, as proposed in other quality-aware systems, we may need to provide alternative versions of the truth (possible worlds) as opposed to simple uncertainty measures. To this end, confidence representation can take these two additional forms:

- Alternative values (or numeric ranges) for a cell.
- Links among tuples that denote that the set might correspond to a same entity (i.e., that they *could* be merged into one).

Dimension	Measure	AssociatedData	Desc.	Author	Date	...
Vertical incompleteness	30%	references to a table and a column	...		04/05/2008	...
Horizontal incompleteness, batch unavailability	20%	data source, batch identifier, missing time window	...		03/05/2008	...
Confidence	90%	references to a table, a column and a cell	...		05/05/2008	...
Consistency	95%	select Diagnosis from Diagnoses where Doctor="John Smith" and Date < 1 1-2008	...	Peter	10/07/2008	...
Confidence, entity resolution	80%	references to a pair of tuples candidates for merging	...	John	11/05/2008	...
...	...	...	...	...	...	...

Figure 4: Example quality metadata to be stored in the DW.

3.3 Warehouse quality metadata

We now briefly describe the metadata that we have to store in the warehouse in order to capture the various aspects of data quality that we outlined above.

There are three aspects we need to capture when attaching quality metadata to the raw data in the warehouse: i) the quality problem (the metric and the measure), ii) the identification of the cells or tuples to which the metric and measure apply, and iii) descriptive information such as why a certain statement on data quality is made, when it was entered, by whom, and so on.

The first aspect is characterized by

1. The *quality dimension*, which specifies what aspect of the data quality we are focusing on. This is one of the quality dimensions we introduced in the previous section.
2. The *measure* for the selected quality dimension. This is a value (could be a percentage, a range, or a binary value) that reflects to which degree there is a data quality problem. It can also be expressed as a function, as discussed next.

The second aspect refers to associating these metadata with the raw data at different granularities, that is, at the levels of individual cells or tuples, as well as entire columns or tables. There are two ways for making this association, namely, extensional and intensional. In extensional, we have to make explicit associations of specific metadata to individual pieces of data in the warehouse. On the other hand, intensional allows us to map specific metadata to a set of data in the warehouse. This mapping may be expressed using a function, and for example an SQL query, therefore providing a large degree of flexibility and control [39]. Figure 4 illustrates the described ideas (albeit, at the conceptual level).

Last, we also attach some descriptive information that can help the analyst to further investigate the causes and consequences of certain quality problems. Examples of such information may be the reason for some data quality problem, explanatory notes, the author of the rule, the date the rule was added, and so on.

4. REPORT QUALITY

In the previous sections, we have described data quality with respect to the raw data in the warehouse. Now we show how quality issues in the base tables of the warehouse affect the quality of the reports presented to users, and how we can interact with the user to inform or get feedback about report quality. Although it is commonly recognized that in BI there is a strong correlation between the quality of data and the quality of business decisions [12], [30], we believe that explicitly assigning quality values to reports as a way to communicate the risk of low-quality decisions has not yet been investigated adequately. Doing so requires us to address the following problems:

1. First, we need to define what a *report* is, and which the different types of reports we need to consider are, in order to better understand the report quality problems.
2. Once we know the different quality problems that we need to model at the base data and at the report level, we need to understand how to *compute report quality* from base data quality, that is, how to populate report quality metadata from base quality metadata.

In the following we discuss these issues. In the next section we instead discuss user interaction with the reports and the personalization of quality metadata.

4.1 Reports and report types

To get and analyze data from a DW, users employ reporting tools that are able to render various reports. Usually reports are defined as a combination of i) methods on how to obtain the data; ii) formatting and layout information for the rendering of the data; iii) properties of reports (e.g., its title, a textual description, the legend). The following are examples of reports:

- *Standard report*: table-based representation of data queried from the DW;
- *Chart report*: graphical representation of data queried from the DW;
- *Pivot report*: allows the comparison of two different data sets against each other; it helps to discover data correlations;
- *Comparison report*: shows differences in a data set considered at two different instants of time.

In the following, we assume that reports are essentially queries over the DW, rendered in some form (tabular or graphical). We therefore assume that a report represents a set of views over the base data. Such views can be computed on the fly or at specified time points, usually (but not necessarily) right after the completion of a new batch load of the warehouse. We consider two types of views: non-aggregated views and aggregated views. *Non-aggregated views* are essentially tables or charts that show raw data from the warehouse, *aggregated views* are tables or charts that show aggregated data (for the purpose of this paper, these are essentially queries with a *group by* statement in them).The above division will help us analyze quality mappings later in this section. In the following, we will use the terms *view* and *report* interchangeably.

Since reports are tables, the quality metadata that describe reports are of the same nature as the one describing base tables. Hence, in general, the problem that we try to solve is to find out, given a set of base tables, the quality metadata on these tables, and a query that defines a view over them, how to map quality metadata on the base tables into quality metadata associated with the view. We do not solve this problem here, but we discuss issues and identify which quality problems at the base data level map into quality problems at the report level, taking in particular into account the issue of aggregations that are common in reports.

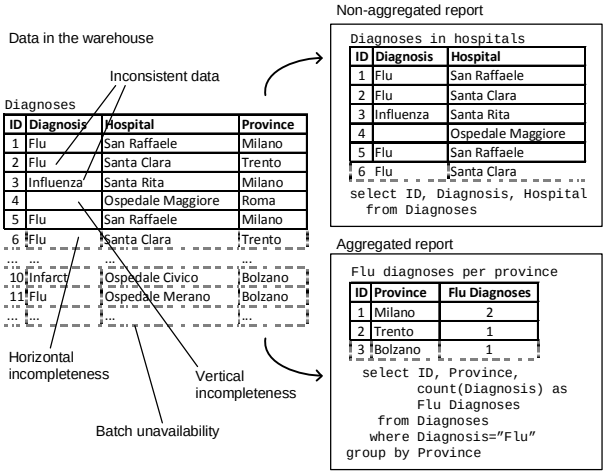
4.2 From Data Quality to Report Quality

In order to understand which quality properties are relevant to characterize reports, we investigate how base data quality affect s the quality of final reports. That is, we look at completeness, consistency, and confidence of base data and derive similar quality properties for reports. For a better understanding, we discuss separately the cases of non-aggregated and aggregated reports.

Data(in)completeness in non-aggregated reports is carried over from the base data to the final report. Therefore, missing cells (vertical incompleteness) or missing tuples (horizontal incompleteness) in the base data result into missing cells or tuples in the report, i.e., into an *incomplete* report. Figure 5 graphically depicts the described scenario: the base data at the left present all three forms of incompleteness (horizontal and vertical and batch unavailability; we will focus on the inconsistency problem next); therefore, the non-aggregated report at the right misses the diagnosis for the "Ospedale Maggiore" and the second tuple for the hospital "Santa Clara", just like the base data; also, no data about the Province of Bolzano can be shown, as the whole respective batch is not available in the data. Notice that not all incomplete tables map into incomplete reports, as the report might select a portion of the base table that is complete. In general this applies to all quality dimensions, and relates to the problem of identifying which base table metadata maps into report quality metadata.

For aggregated reports, data incompleteness may lead to missing cells in the report only if in the base data all the values of a group are missing (e.g., used to calculate a sum). In Figure 5, for instance, the aggregated report lacks the diagnoses for the Province of Roma, due to the missing diagnosis in the base data (actually, we don't know whether it should be in the report or not, as we do not know which exact diagnosis value is missing for that Province). The batch unavailability of the data from the Province of Bolzano, on the other hand, should be in the aggregated report, but the respective data is missing in the DW. If instead an aggregated value is computed over a column/attribute with only partially missing data, a value can be computed and, hence, the report is not incomplete. In this case, we can say that the incompleteness of the base data affects the *confidence* of the final report. This is exemplified in Figure 5 by the tuple regarding the Province of Trento (we should actually see 2 flu diagnoses), which is computed over incomplete data; the tuple regarding the Province of Milano is correct.

Data consistency problems in the generation of non-aggregated reports will carry over from the base data to the reports. That is, misunderstandings of the semantics of fields and different abstraction levels or units will unavoidably show up in the report as inconsistent data, just like they are in the base data. For instance, the non-aggregated report in Figure 5 presents the same inconsistencies as its base data, i.e., "Flu" vs. "Influenza". For aggregated reports, data consistency problems typically lead to low *report confidence*: if aggregated values (e.g., the number of flu diagnoses per Province in Figure 5) are computed over a column with inconsistency problems, the final result will be characterized by a low confidence, as we cannot be sure whether all relevant values have been considered in the computation or not. Indeed, there would be 3 flu diagnoses for the Province of Milano, but the hospital "Santa Rita" has entered "Influenza" instead of "Flu", the respective tuple could not be counted. The confidence of that data for the Province of Milano is hence low.

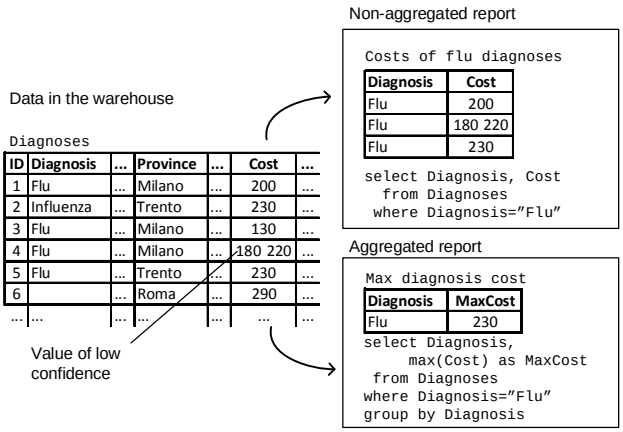


ID	Diagnosis	Hospital	Province
1	Flu	San Raffaele	Milano
2	Flu	Santa Clara	Trento
3	Influenza	Santa Rita	Milano
4		Ospedale Maggiore	Roma
5	Flu	San Raffaele	Milano
6	Flu	Santa Clara	Trento
...	...	...	...
10	Infarct	Ospedale Civico	Bolzano
11	Flu	Ospedale Merano	Bolzano
...	...	...	...

ID	Diagnosis	Hospital
1	Flu	San Raffaele
2	Flu	Santa Clara
3	Influenza	Santa Rita
4		Ospedale Maggiore
5	Flu	San Raffaele
6	Flu	Santa Clara

ID	Province	Flu Diagnoses
1	Milano	2
2	Trento	1
3	Bolzano	1

Figure 5: Effects of data completeness, batch unavailability, and data consistency on report quality. The SQL queries show how the reports are computed. Dashed lines represent expected, but not complete or available data.



ID	Diagnosis	...	Province	...	Cost	...
1	Flu	...	Milano	...	200	...
2	Influenza	...	Trento	...	230	...
3	Flu	...	Milano	...	130	...
4	Flu	...	Milano	...	180 220	...
5	Flu	...	Trento	...	230	...
6		...	Roma	...	290	...
...	...	...	...	...	...	...

Diagnosis	Cost
Flu	200
Flu	180 220
Flu	230

Diagnosis	MaxCost
Flu	230

Figure 6: Effects of data confidence on report quality. Note that the value "180-220" is intended as an indication for low confidence in absence of a precise specification of quality metadata.

Data confidence properties carry over in the computation of non-aggregated reports and directly affect the *report confidence*. In non-aggregated reports, data values with low confidence just carry over, resulting in a report that includes values with low confidence. Figure 6 depicts for example how the cost value "180-220" carries over from the base data to the non-aggregated report, maintaining its low level of confidence. The same is true for aggregated reports as well, where aggregated values with low confidence may lead to an aggregated value of low confidence.

However, in some cases aggregated reports may eliminate the lack of confidence originating from the base data. Consider the following example, depicted in Figure 6. Assume we need to create a report and compute the maximum cost for flu diagnoses out of the table in Figure 6, which presents one value ("180-220") with low confidence. Even though there is an evident confidence problem, the report will contain the correct result (i.e., "290"), which will also be accurate.

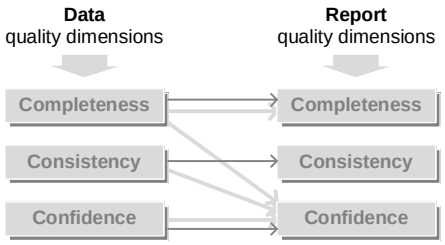


Figure 7: Mapping of raw data quality properties into report quality properties. Dark-gray arrows refer to non-aggregated reports, light-gray arrows to aggregated reports.

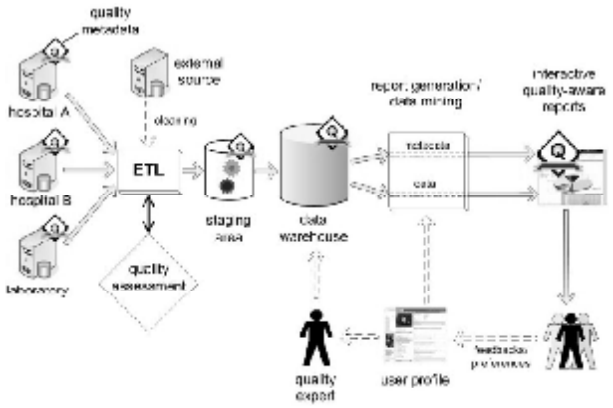


Figure 8: Quality-aware reporting and user quality profiles for the fine-tuning of quality metadata (the "Q" rhombuses).

Figure 7 graphically summarizes the above discussion on how report quality is determined by raw data quality. The dark-gray arrows in the figure represent the mapping for data quality properties into report quality properties for non-aggregated reports; light-gray arrows represent the mapping for aggregated reports.

5. USER INTERACTION WITH QUALITY-AWARE REPORTS

Once we have a quality metadata framework for reports as discussed above and a way to compute report quality metadata, we can use this information to visualize quality-aware reports and support user interaction with them, as well as leverage quality information for the analytics algorithm developed on top of the warehouse or on top of the reports. Specifically, we envision the following opportunities (and consequent research challenges):

- *Visualization*: How to visualize quality information in a way that is easy to "consume" and understand if and which parts of the report are meaningful and can be used to take business decisions. This aspect also has a degree of "subjectiveness" and therefore includes taking into considerations user preferences and quality profiles, both in terms of how to show information and, from a more semantic viewpoint, to include users' personal beliefs on data quality.
- *Interaction*: How to have the user interact with the report in 3 ways: i) to "simulate" alternative view reports based on varying assumptions on the data quality and based on what to consider for the report computation (e.g., confidence thresholds); ii) to define personal quality profiles in the meaning described above; iii) to provide feedback on the quality assumptions to be used by the system to correct data cleaning procedures or quality metadata computation procedures, so that the user's knowledge can be used not only for the subjective report views but also as "objective" information for the benefit of all report consumers.
- *Analytics*: This aspect is related to data and reports as consumed by applications (and typically by BI applications). The techniques here are application specific, so in this paper we state some general issues related to quality-aware business intelligence.

In the following, we detail research issues and preliminary ideas related to these topics. We begin by discussing the quality profiles which is the user-specific metadata common to all the above issues. Then we briefly discuss research issues in the areas of visualization, interactions, and analytics over quality-aware reports.

5.1 Quality profiles

Quality profiles are user-specific report configuration data that define (i) the *appearance* of quality metadata (how the report is graphically tagged) and (ii) how to *filter* and adjust imperfect data that contributes to the computation of a report.

We refer to the latter information as *quality tuning* metadata. We distinguish between *general* tuning data and *report-specific* tuning data. The first include user beliefs that are applicable across all reports (e.g., "I always trust data from St. John's"). The latter includes tuning of a specific report model (e.g., on the monthly report on the average cost of surgical procedures by unit) or even a report instance (the above report computed for June 2008).

Figure 8 extends the scenario architecture introduced in Figure 1 with user/quality profile metadata, and highlights how user feedback and preferences may drive the report generation and data mining processes. Collected tuning metadata may also be assessed by a quality expert, possibly propagating feedback into the actual quality metadata in the DW.

The tuning metadata can include this information:

- At the simplest level, tuning can simply mean having a personalized version of the quality metadata. This implies that the end user can view and edit, e.g., the completeness or confidence values, or even the intensional descriptions of the quality metadata. (Note that we are not concerned here with the UI and in general the user support for editing such metadata easily, but just in the end results, that is, the quality profile). This "rewriting" can be applied to some or to all original metadata entries, and can also include new entries not originally captured (e.g., the warehouse may believe that data from St. John is complete but the end user may know that this is not the case). It also applies to metadata specific to a report, as well as to general metadata, so that for example the user can also state that while the warehouse metadata states that St. John's data is in general uncertain, from their perspective it is certain.

- Users can define threshold levels and metadata aggregation functions (as e.g. in [8]) for data to be included in the report computation. For example, it is possible to express that, although the quality values are not changed, only data with a quality level above a threshold are included.
- The above approach can be generalized by having end users define a metadata policy that determines which metadata and which data are to be considered in the generation of the (personalized) report. In general, users can define two kinds of queries over the quality metadata to express this behavior: the first is *metadata filtering functions*, that is, a set of queries or procedures that outputs which metadata entries should be considered. For example, one can decide to only consider quality metadata entries by herself only or by a quality metadata manager she trusts. This is analogous to an "intensional tuning", where the tuning is specified by using functions. The second is *data filtering functions*, that is, intensional definition of functions that define thresholds for data to be included in the report (e.g., never include data that is less than 30% complete).
- The final category includes *replacement functions*, that is, definition of algorithms to replace data with quality problems with estimates. For example, users may decide to replace incomplete data with estimates from the previous month.

5.2 Visualization and interaction

Visualizing personalized quality information and letting users interact with such information to modify the report based on their perceived quality is a key goal of this line of research. In general visualization, especially when end users are involved, is a complex issue as a poor visualization paradigm may defeat the purpose of the entire work.

Any visualization approach needs to be accompanied by an HCI study to assess its understandability. The development and assessment of the visualization paradigm is in progress; here we limit ourselves to some key aspects and discussion points.

Just like (graphical) report design is a human-intensive activity and cannot be easily demanded to automated generation if we want a usable result, the same holds for quality metadata. While we can develop default representations for the various quality issues identified earlier, it should be possible to tailor the final results to the users and to the nature of the problem. Hence, we expect the quality-aware report design infrastructure to allow for report design of quality aspects (based on a set of primitive quality visualization concepts) as well, rather than imposing a default visualization. In our research, we focus on which these quality visualization primitives are and on how they can be combined in a quality-aware report.

On the interaction side, the main issues to be addressed and functionality to be provided are the following:

Interactive quality exploration: Users should be able to "play" with the quality information and preview what the report results would be if quality metadata were different. For example, Figure 9 represents different projections of the same chart that will be changed depending on the selected confidence level (in the figure denoted by the slider position). Depending on such a parameter, relevant data will be considered to create a chart that will be updated on the fly. Trying to select various levels of confidence, the user can see for instance the best/worst case of the spreading of a particular disease, with certain extent of confidence in such a fact.

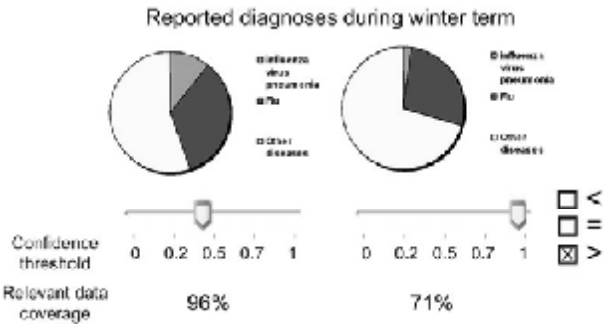


Figure 9: Interactive quality-aware report.

The percentage of relevant data coverage is shown under the chart for which the data was taken. The value shows the user how much data is taken into consideration when constructing the chart and how much data with lower confidence is left out of the analysis.

Completeness problems may also be addressed by removing tuples with incomplete data or by inserting missing values, e.g., by extrapolating/predicting data from past values.

In general, we envision the following ways for interacting with report quality metadata: users should be able to "turn off" some quality problems (e.g., assume the information is correct and complete and disregard the quality metadata entry), to change the quality measures, to edit the logic of intensional rules (e.g., maintain the rule that data from St. John is low confidence, but exclude the case of Dr. Hyde who is known to be reliable), or to compensate for the quality problems with custom logic (e.g., predicting missing values). All this requires visualization primitives and, for the case of intensional rule editing, requires approaches similar to Query By Example (QBE) [29]. Again, the interaction for exploration purposes may be based on defaults or it may be designed ad hoc. For example, for editing an intensional rule, we can either provide a generic QBE paradigm, or we can provide a simplified interface, designed ad hoc for a given report or rule, where for example users can explicitly define / modify intentional rules based on doctors or hospital of provenance.

Finally, to support all of the above, we also need to let the user know the reasons behind the quality metadata measures, that is, why the values are the way they are, based on what assumptions, and who defined these assumptions.

Quality feedback: The result of the exploration can be stored either as part of the personalized quality profile, or it can be proposed as a generally applicable rule that modifies the report metadata and possibly also the base quality data. This topic presents challenges ranging from how to map report metadata changes back into base metadata changes, to how to assess the validity of the proposed changes. Analogously to approaches on case based reasoning, here we need to assess the quality of the feedback and decide 1) how to embody it into the metadata so that it can be shown to different end users to make them aware that other users have made different assumptions, and 2) how to understand when to combine various feedbacks and propose them to analysts for incorporation into the ETL and quality measurement processes.

5.3 Analytics

The components of the solution we presented above are focused on the interaction of the quality-aware data warehouse with the human users of the system. Another aspect of the proposed approach that is equally important is the introduction and use of data analysis and data mining techniques that are quality-aware.

Most of the current data mining techniques assume that the data they operate on are complete and accurate. Therefore, they produce exact, deterministic results based on these data values. This is not true for privacy preserving data mining, where the objective is to perturb the original values while producing a correct final result (see [40] for a brief overview of the work in this area). In our case we face a different problem, namely, how to analyze and mine data that are inherently not exact and complete. The system we are proposing will be coupled with quality-aware data mining algorithms, which will treat all the quality metadata as first class citizens. These metadata will be used by the data mining models, along with the data values, in order to produce the final results. This way, the mining results will incorporate knowledge of the inaccuracy/incompleteness of the original data and will expose to the users a range of possible answers with confidence values characterizing their accuracy. Recent studies have looked at similar problems in the areas of OLAP [8] and deviation detection [1].

6. RELATED WORK

Several studies have focused on the problem of data quality. They have examined different aspects of this problem, ranging from the definition of the term "quality" in this context, to the management and processing of all the quality-related information, and in such diverse domains as health care [19] and manufacturing [28].

6.1 Data quality

The problems of characterizing and dealing with data quality have been the focus of several studies. Data quality problems emerge in literally every application domain, and techniques for addressing such problems have been proposed in the literature.

Data quality can be measured and quantified according to various parameters. Previous works provide different classifications of the data quality dimensions [35], [20], [6], [17], [26]. These classifications provide a basic set of data quality dimensions, among which accuracy, completeness, consistency, interpretability, timeliness, and understandability [32]. However, defining objective measures for all the above dimensions is a challenging task, and an active area of research.

Data quality has also been identified as an important problem in other domains as well, such as manufacturing, and business processes. In this context, several approaches have been proposed for managing data quality problems, with the most prominent one being the Six Sigma approach [28].

6.2 Data provenance

When data are stored in some database, we are interested in keeping track of information related to the provenance of these data [36]. An important issue in data providence is its characterization. That is, to find the answers of questions like "*why is a piece of data in the output?*" and "*where is the piece of data copied from?*" Buneman et al. [7] target these issues and propose a framework for describing and understanding provenance. The more recent work of Green et al. [15] describes provenance in the context of incomplete and probabilistic databases. In [24] the authors specifically focus on provenance quality data in scientific workflows. Provenance metadata may also play a major role in assigning and managing quality measures.

Sometimes the propagation of annotations is dependent on the syntax of the query. One may want to control the propagation of annotations in a schema. The custom propagation schemes allow the user to specify where to obtain annotations from. Bhagwat et al. [5] present propagation schemes that are essentially based on where data is copied from.

6.3 Identity resolution

Another problem relevant to quality is that of identity resolution or duplicate detection (i.e., whether two different pieces of data refer to the same real world object).

Duplicate detection through record linkage has been extensively studied [11]. Many of these approaches are based on different flavors of clustering algorithms. A clustering technique is also the basis of the approach proposed by Andritsos et al. [1]. Using rule-based approaches [21], [14], it is easier to create a large number of training pairs that are either clearly non duplicates or clearly duplicates. Despite that, the rule-based approaches require user intervention in rule management scenarios. Recent approaches have also focused on the problem of how to efficiently support the duplicate identification operation in the context of relational database systems [16].

Several data cleaning techniques have also been proposed for the problem of structural heterogeneity (for example, representing a date as *year/month/day* in place *day/month/year*, or the location of a room as *room number-building-university* in place of *university-room number-building*) [31].

6.4 Uncertainty in databases

Problems related to data uncertainty have been studied in the past in the area of databases and data warehouses. Several studies have proposed a framework for quality-oriented data warehouse design [18], [38], [37], [12]. These frameworks take into account the entire lifecycle of the data warehouse, and are able to track the quality of data at each stage of the process. The above approaches aim at setting a quality goal, evaluating the current quality status, and finally at analyzing and improving the current situation. The same principles have also been applied to the domain of health care data [19], where a process model for the data warehouse lifecycle of health care data is described, that is able to capture errors in the design, integration, and use of the warehouse. Nevertheless, none of the above studies focuses on the specific problems relevant to report generation, use, and management, when quality measures are taken into account.

Recently, there has been lots of interest in databases specifically designed to manage uncertain data [3], [4], [34], [10]. In this case, data are coupled with a probability value indicating the degree of confidence to the accuracy of the data. These probabilities are then taken into account by the database management system when processing the data to produce answers to user queries. The difference to our approach is that the above systems do not deal with the problems of assigning these probabilities and of deriving them in complex cases, such as when computing reports. In this case, we need to reason about quality measures that are assigned to objects of different granularities (e.g., cells, tuples, or tables), and we also need to use semantics as to how to combine the different quality measures.

7. CONCLUSION AND OUTLOOK

In this paper we have investigated an end-user-centric business intelligence view on the problem of low data quality, proposing what we call *quality-aware business intelligence*. We have discussed how low quality data in input affects the quality of the output of a business intelligence application, i.e., the reports. Accordingly, we have proposed the use of *quality-aware reports*, allowing the end-users to interactively "play" with report quality metadata, finally enabling them i) to be aware of the quality of the report they are looking at, ii) to fine-tune a report based on personal knowledge about the quality of the underlying data, and iii) to provide and share with other users quality-related feedback.

In this study, we have highlighted novel challenges and open issues in handling low quality data in business intelligence applications, which we believe will play a major role in business intelligence over the next years. We are currently pursuing the research directions outlined in the previous sections at both the warehouse and the report levels (which represent the focus of our work), but we know that we have only touched on the full problem and that there are plenty of related issues that still demand an answer.

Acknowledgements

We would like to thank Cinzia Cappiello for her valuable comments and suggestions. This work was supported by funds from the European Commission (contract N° 216917 for the FP7-ICT-2007-1 project MASTER).

8. REFERENCES

[1] C. C. Aggarwal, P. S. Yu: Outlier Detection with Uncertain Data. SDM 2008: 483-493.

[2] P. Andritsos, A. Fuxman, and R. Miller, "Clean Answers over Dirty Databases: A Probabilistic Approach," in ICDE, 2006.

[3] L. Antova, C. Koch, and D. Olteanu, "10^10^6 Worlds and Beyond: Efficient Representation and Processing of Incomplete Information," in ICDE, 2007, pp. 606-615.

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom, "Databases with uncertainty and lineage," VLDBJ, vol. 17, no. 2, p. 243–264, 2008.

[5] D. Bhagwat, L. Chiticariu, C. W. Tan, and G. Vijayvargiya, "An annotation management system for relational databases," VLDBJ, vol. 14, no. 4, pp. 373-396, 2005.

[6] M. Bovee, R. P. Srivastava, and B. Mak, "A Conceptual Framework and Belief Function Approach to Assessing Overall Information Quality," in IQ, 2001, pp. 311-328.

[7] P. Buneman and S. Khanna, "On Propagation of Deletions and Annotations through Views," in PODS, 2002.

[8] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, S. Vaithyanathan: OLAP over uncertain and imprecise data. VLDB J. 16(1): 123-144 (2007).

[9] C. Cappiello, C. Francalanci, and B. Pernici, "Data Quality Assessment from the Users Perspective," in IQIS, 2004, pp. 68-73.

[10] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," VLDBJ, vol. 16, pp. 523-544, 2007.

[11] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," IEEE Transactions on Knowledge and Data Engineering,, vol. 19, no. 1, pp. 1-16, 2007.

[12] L. P. English, Improving Data Warehouse and Business Information Quality. John Wiley & Sons, 1999.

[13] A. Even and G. Shankaranarayanan, "Understanding Impartial Versus Utility-Driven Quality Assessment in Large Data-sets," ICIQ'07.

[14] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita, "Declarative Data Cleaning: Language, Model, and Algorithms," in VLDB, 2001, pp. 371-380.

[15] T. J. Green, G. Karvounarakis, and V. Tannen, "Provenance Semirings," in PODS, 2007.

[16] S. Guha, N. Koudas, A. Marathe, and D. Srivastava, "Merging the Results of Approximate Match Operations," in VLDB, 2004, pp. 636-647.

[17] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis, "Architecture and Quality in Data Warehouses: An Extended Repository Approach," Information Systems, vol. 24, no. 3, pp. 229-253, 1999.

[18] M. Jarke and Y. Vassiliou, "Data warehouse quality: a review of the DWQ project," in IQ, 1997, pp. 299-313.

[19] R. L. Leitheiser, "Data Quality in Health Care Data Warehouse Environments," in HICSS, 2001.

[20] A. Levitin and T. Redman, "Quality Dimensions of a Conceptual View," Information Processing and Management, vol. 31, no. 1, pp. 81-88, 1995.

[21] E. Lim, J. Srivastava, S. Prabhakar, and J. Richardson, "Entity Identification in Database Integration," in IEEE International Conference on Data Engineering, 1993, p. 294–301.

[22] P. Missier, S. Embury, M. Greenwood, A. Preece, and B. Jin, "Quality views: capturing and exploiting the user perspective on data quality," in VLDB, 2006.

[23] P. Missier, S. Embury, M. Greenwood, A. Preece, and B. Jin, "Managing Information Quality in e-science: the Qurator workbench," in SIGMOD, 2007.

[24] P. Missier, S. Embury, R. Stapenhurst. "Exploiting provenance to make sense of automated data acceptance decisions in scientific workflows", IPAW'08, Salt Lake City, USA.

[25] F. Naumann, J. C. Freytag, and U. Leser, "Completeness of integrated information sources," Information Systems, vol. 29, pp. 583-615, 2004.

[26] F. Naumann, Quality-Driven Query Answering for Integrated Information Systems. Springer, 2002.

[27] T. Palpanas, N. Koudas, and A. Mendelzon, "Using Datacube Aggregates for Approximate Querying and Deviation Detection," IEEE transactions on knowledge and data engineering, vol. 17, no. 11, pp. 1465-1477, 2005.

[28] T. Pyzdek, The Six Sigma Handbook, Second ed. McGraw-Hill, 2003.

[29] R. Ramakrishnan and J. Gehrke, Database Management Systems, 3rd ed. McGraw-Hill, 2002.

[30] T. C. Redman, Data Quality for the Information Age. Artech House, 1996.

[31] S. Sarawagi, "Special Issue on Data Cleaning," Bulletin of the IEEE Technical Committee on Data Engineering, vol. 23, no. 4, 2000.

[32] M. Scannapieco and T. Catarci, "Data Quality under the Computer Science Perspective," Archivi & Computer, vol. 2, 2002.

[33] M. Scannapieco and C. Batini, "Completeness in the Relational Model: a Comprehensive Framework," in IQ, 2004, pp. 333-345.

[34] S. Singh, et al., "Database Support for Probabilistic Attributes and Tuples," in ICDE, 2008, pp. 1053-1061.

[35] D. M. Strong, Y. W. Lee, and R. Y. Wang, "Data Quality in Context," Commun. ACM, vol. 40, no. 5, pp. 103-110, 1997.

[36] W. C. Tan, "Provenance in Databases: Past, Current, and Future," IEEE Data Eng. Bull., vol. 30, no. 4, pp. 3-12, 2007.

[37] D. Theodoratos and M. Bouzeghoub, "Data Currency Quality Factors in Data Warehouse Design," in DMDW, 1999.

[38] P. Vassiliadis, M. Bouzeghoub, and C. Quix, "Towards Quality-oriented Data Warehouse Usage and Evolution," Information Systems, vol. 25, no. 2, pp. 89-115, 2000.

[39] D. Srivastava, Y. Velegrakis: Intensional associations between data and metadata. SIGMOD Conference 2007: 401-412.

[40] V. S. Verykios, et al., "State-of-the-Art in Privacy Preserving Data Mining," ACM SIGMOD Record, vol. 3, no. 1, pp. 50-57, 2004.

[41] R. Y. Wang and D. M. Strong, "Beyond accuracy: what data quality means to data consumers," Journal of Management Information Systems, vol. 12, no. 4, pp. 5-33, 1996.