EFFICIENT PROFILING FOR ESTIMATION OF QUERY RESULT QUALITY

Naiem K. Yeganeh,
University of Queensland, Australia
naiem@itee.uq.edu.au

Shazia Sadiq
University of Queensland, Australia
shazia@itee.uq.edu.au

Mohamed A. Sharaf
University of Queensland, Australia
m.sharaf@uq.edu.au

Ke Deng
University of Queensland, Australia
k.deng@uq.edu.au

Abstract: The issue of Data Quality (DQ) is of increasing importance as individuals as well as corporations are relying on multiple, often external sources of data to make decisions. Data quality profiles consist of statistical measurements about the quality of data sets. Query systems can use DQ profiles as a form of metadata to estimate the quality of a query result set. Traditional DQ profiling provides an estimate on the overall quality of a data set or data source, but quality of a query result can be remarkably different from the overall quality of the data set because conditions within the query typically select a subset of the data. In this paper we propose an efficient conditional DQ profiling method which can estimate the quality of a result set for a given query with guaranteed user definable level of accuracy.

Key Words: Data Quality Profiling, Conditional Data Quality, Query Result

1. INTRODUCTION

User satisfaction from a query response is a complex problem encompassing various aspects including efficiency as well as the quality of response. Quality in turn includes several dimensions such as completeness, currency, accuracy, relevance, etc. In current information environments where individuals as well as organizations are routinely relying on multiple, external data sources for their information needs, absolute metrics for data quality are no longer valid. Thus the same data set may be valuable for a particular usage, but useless for another.

Consider for example a virtual store that is compiling a comparative price list for a given product (such as Google products, previously known as froogle) through a meta-search (a search that queries results of other search engines and selects best possible results amongst them). It obviously does not read all the millions of results for a search and does not return millions of records to the user. It normally selects top *k* results (where *k* is a constant value) from each search engine and finally returns top *n* results after the merge.

In the above scenario, when a user queries for a product, the virtual store searches through a variety of data sources for that item and ranks and returns the results. For example the user may query for “Canon PowerShot”. In turn the virtual store may query camera vendor sites and return the results. The value that the user associates with the query result is clearly subjective and related to the user's intended

requirements, which go beyond the entered query term, namely "Canon PowerShot" (currently returns 150,000 results from Google products). For example the user may be interested in comparing product prices, or the user may be interested in information on latest models.
More precisely, suppose that the various data sources can be accessed through a view consisting of columns ("Item Title", "Item Description", "Numbers Available", "Price", "Image", "User Comments"). Two different users searching for "Canon PowerShot" may have two different interests: One may be interested to learn about different items (products) and may not care about the "Numbers Available" and "Image" columns. However, consistency of "Item Title" and completeness within the populations of "User Comments" in the query results, are more important. Another user may be sure about the item to purchase but be searching for the best price. Obviously "Price" has the greatest importance for this user. It should be current and accurate.
Above example highlights the importance of DQ considerations for query answering. Indeed, satisfaction of the user from query results is highly related to the perceived quality of the result of given query. Thus, a query system should be able to access statistics about different data sets to estimate the quality of the result set of the query against each data source. For example if data source S_1 has high quality data on price for "Cannon" cameras, a data quality aware query system (DQAQS) should have a mechanism to rank S_1 higher than other data sources if the quality of price is important to a user querying for "Cannon" cameras.

Item Title	Item Desc	Num Available	Price	Image	User Comments
S2IS	Canon PShot	8	310	1.jpg	
	Camera		1000		
S3IS	Canon PShot		375		<4 Reviews>
XL-2		4	184	a.bmp	
DMC-TZ5K	Panasonic Lmx		340		
DSC-W55	S ony Cshot	2	260	9.jpg	

(a)

Object	Metric	Value
Shop.ItemTitle	Completeness	0.83
Shop.ItemDesc	Completeness	0.83
Shop.NumAvailable	Completeness	0.50
Shop.Price	Completeness	1.00
Shop.Image	Completeness	0.50
Shop.User Comments	Completeness	0.16

(b)

Figure 1 A traditional data quality profile for a sample dataset.

Measurement of the data quality of a data set is called DQ profiling. Overall statistical measurements for a data set - which we call traditional DQ profile - as shown in Figure 1 is widely used in research literature and industry [13][9]. Figure 1 (b) represents a traditional DQ profile for the given data set of Figure 1 (a). DQ profile of Figure 1 (b) is the result of measuring completeness of each attribute in the dataset presented in Figure 1 (a). In this example completeness is considered as the number of null values over the number of records in the data for each attribute. For example, completeness of the "Image" attribute of the given data set is 50% since there are 3 null values over 6 records. The three columns in the profile table of Figure 1 (b) identify object (an attribute from the relation) against which the metric Completeness is measured as well as the result of this measurement.
Given the DQ profile of Figure 1 (b) as the profile table for data source S_1, a query engine can estimate that the quality of the attribute "Image" resulting from data source S_1 will be about 50% and can use this information to rank data source S_1 based on the projected quality of the result from a given query. In [13] a service-oriented architecture for DQ aware query systems is proposed in which a DQ profiling service is deployed for generation of DQ profiles from data sets which utilizes traditional DQ profiling.
Traditional DQ profiles which are similar to the one in Figure 1 (b) are incapable of returning reliable estimates for the quality of the result set when results are bounded with conditions. For example; even though the completeness of the Image attribute for data source S_1 is 50%, the completeness of the Image attribute for query results (with conditions) from this data source can be anything between 0% and 100%: Completeness of the Image attribute for "Cannon" cameras is 50%, this value is 100% for "Sony" cameras, and 0% for "Panasonic" cameras. In reality, a Cannon shop may have records of other brands in their database, but they are particularly careful about their own items, which means the quality of Cannon items in database will significantly differ from the overall quality of the database.

In this example, the selection conditions (brand is “Cannon” or brand is “Sony”) are fundamental parts of the query. However, they have not been adequately considered in previous work on data quality profiling. This paper introduces a novel extension of traditional DQ profiles, referred to as conditional DQ profiles, that is capable of correct estimation of the quality of query results. Conditional DQ profile consists of a set of: Conditions → DQ measurements, where Condition refers to a query’s selection condition (like the WHERE clause in SQL) and DQ measurement infers the quality of the selection query result (e.g. Brand=Cannon →Completeness=50%). Traditional DQ profile is a special case of conditional DQ profile where the condition is empty. Basically, benefit of proposed algorithms versus traditional profiling technique is that the proposed algorithms are able to accurately estimate the quality of any given query with equality selection conditions, but estimation of traditional DQ profile is only valid for the whole dataset.

Definition and modelling of conditional DQ profile, is the first contribution of this paper. Brute-force generation of DQ profile is easy but exhaustively expensive for both creation time and storage costs. Second contribution of this paper is to present algorithms to generate conditional DQ profile with minimum overhead, but with guaranteed accuracy for estimation of DQ results.

Rest of the paper is organized as follows: Related works are studied in Section 2. In Section 3 we first formulate the problem and study the brute-force approach, and then we propose two algorithms for efficiently generating a near minimal conditional DQ profile. In Section 4 we suggest a method to estimate the quality of query results using conditional DQ profile. In Section 5 we evaluate our algorithms and finally in Section 6 we conclude and discuss future works.

2. RELATED WORKS

Consequents of poor quality of data have been experienced in almost all domains. From the research perspective, data quality has been addressed in different contexts, including statistics, management science, and computer science [10]. To understand the fundamental concepts of data quality, various research works have defined a number of quality dimensions [10] [12].

Data quality dimensions characterize data properties e.g. accuracy, currency, completeness, etc. Many dimensions are defined for assessment of quality of data that give us the means to measure the quality of data. Data Quality dimensions can be very subjective (e.g. ease of use, expandability, objectivity, etc.).

To address the problems that stem from the various data quality dimensions, the approaches can be broadly classified as investigative, preventative and corrective. Investigative approaches essentially provide the ability to assess the level of data quality and are generally provided through data profiling tools. Many sophisticated commercial profiling tools exist [5]. There are several interpretations of dimensions which may vary for different use cases. For example, Completeness may represent missing tuples (open world assumption) [1]. Accuracy may represent the distance from truth in the real world. Such interpretations are difficult if not impossible to measure through computational means and hence in the subsequent discussion, the interpretation of these dimensions is assumed as a set of DQ rules.

A variety of solutions have also been proposed for preventative and corrective aspects of data quality management. These solutions can be categorized into following broad groups: Semantic integrity constraints [3]. Record linkage solutions, which has been addressed through approximate matching [6], de-duplicating [7] and entity resolution techniques [2]. Data lineage or provenance solutions are classified as annotation and non-annotation based approaches where back tracing is suggested to address auditory or reliability problems [11]. Data uncertainty and probabilistic databases are another important consideration in data quality [8].

Measurements made on a dataset for DQ dimensions are called DQ metrics and the act of generating DQ metrics for DQ dimensions is called DQ profiling. Profiling generally consists of collecting descriptive statistical information about data. These statistics can in turn be used in query planning, and query optimization. Data quality profile is a form of metadata which can be made available to the query processing engine to predict and optimize the quality of query results.

Literature reports on some works on DQ profiling. For example in [9] data quality metrics are assigned to

each data source in the form of a vector of DQ metrics and their values. In [12] a vector of DQ metrics and their values is attached to the table's meta data to store additional DQ profiling; e.g. *{(Completeness, 0.80), (Accuracy, 0.75),…}*. In [14] an additional DQ profile table is attached to the relation's metadata to store DQ metric measurements for the relation's attributes.
However, the above profiling techniques are not always sufficient to estimate the quality of query results. In Section 3 we discuss this limitation in detail.

3. CONDITIONAL DATA QUALITY PROFILING

DQ profiling is the task of measuring DQ metrics for given data. In this paper we assume that services are available to measure DQ metrics. In-specific, we do not need to know how *Completeness* or *Consistency* of data is defined; instead we assume a service that calculates DQ metrics for every row of the dataset.
Information collected in DQ profiles clearly affects quality estimation of query results [14]. However, traditional DQ profile does not provide adequate information to for this purpose. The only situation where metric value of the whole dataset will be similar to the metric value of the result of any query over that dataset is when distribution of dirty data within dataset is evenly random. On the other hand, A reasonably good DQ profile must be able to correctly estimate the quality of the query result even in the situation that the query limits the results from the dataset by a set of conditions and distribution of dirty data is not totally uniform within the dataset.
The limitation of traditional DQ profiling motivates us to propose *Conditional DQ Profile* to estimate DQ profile metrics where distribution of dirty data in the dataset is not evenly random and queries are bound with conditions. In order to define Conditional DQ Profile, we first need to formally define DQ metric function.

Definition 1. *Let {a_1, …, a_m} be all attributes of the relation R, T be a set of tuples representing R, and metric m be a set of rules. We define* ***metric function*** $m_{a_i}(t), t \in T$ *as 1, if the value of attribute a_i from tuple t, does not violate any rule in m; and 0 otherwise.*

DQ metric function m_{a_i} is a set of rules, where each rule is a Boolean function that describes DQ dimension. For example, accuracy metric function for a given postcode can be defined by a set of rules containing a check that compares data against some master data.
Assuming conditions in a finite domain consist of single comparisons, combined together with ∧ operators, we define Conditional DQ profile as a table that maps conjunctive conditions of queries to DQ metric values[1]. For every metric functionm_{a_i}, we create a Conditional DQ profile table as a set of conditional DQ profile records, and defined as below:

Definition 2. *Let m_{a_i} be an arbitrary metric function for attribute a_i, and T be set of tuples for relation R. We define conditional DQ profile Pr as a set of DQ profile records t_{Pr}.*

Definition 3. *Let Pr be a conditional DQ profile, consisting of profile record t_{Pr}, and Condition be the set of conjunctive equality conditions for a given query that results in a subset of dataset $\varsigma \subseteq T$. We define t_{Pr} as Condition → ($\#_\varsigma$, Q_ς), where Condition is a query's conjunctive selection condition and ς is a subset of T that satisfy Condition. $\#_\varsigma = |\varsigma|$ and $Q_\varsigma = |\{t \in \varsigma \,|\, m_{a_i}(t)=1\}|$.*

In relational databases, we store the profile table *Pr* for metric m_{a_i} (also referred as $Pr_{m_{a_i}}$) as a relation *(a_1,… , a_n, $a_\#$,a_Q)* consisting of profile row tuples $t_{Pr} = (t_{1_{Pr}}, \dots, t_{n_{Pr}}, \#_{t_{Pr}}, Q_{t_{Pr}})$ where $t_{i_{Pr}} \in dom(a_i) \cup$ "_" (don't care value). $\#_{t_{Pr}}$ is the number of appearances of the tuple in *T* considering "_" which substitutes for anything, and $Q_{t_{Pr}}$ is the number of tuples among them that return *1* for m_{a_i}.

[1] Calculation of DQ profile is independent from ∨ operator since |A∪B|=|A|+|B|-|A∩B|.

If we are given all possible conjunctive equality conditions that can happen for a given dataset, we can create the conditional DQ profile Pr_{a_i}or any given metric function. However, the number of possible conjunctive equality conditions over a given dataset can be exhaustive and the Conditional DQ Profile that would be created from all possible queries can become much larger than the underlying dataset.

Conditional DQ Profile Generation

As described above, the complete search space for a conditional DQ profile may need to traverse all possible conjunctive equality comparisons.

Brute-force Approach: Conditional DQ profile generation incurs the following challenge: Size of a complete conditional DQ profile which has all data required for estimating the quality of any query result, can be as large as the original dataset or even larger. Thus, storing and querying conditional DQ profile may be too expensive and inappropriate.

To illustrate this problem, consider relation *Items(Brand,Model,Price)* in Figure 2 (a) (we also refer as *B,M,P* for simplicity). Values these attributes come from the following domains: *dom(B) = {Sony, Cannon}* (abbreviated as {S,C}), *dom(M)={SLR, Norm}* ({S,N}), and dom(P)={Low, High} ({L,H}). Each domain has 2 members, by addition of "_", there can be 3^3 possible queries. {B=C or B=S or B=_, M=S or M=N or M=_, P=H or P=L or P=_}. Search space consists of all possible subsets of equality comparisons joined with ∧ operator (e.g. {B=C ∧ M=S ∧ P=H}). The number of possible comparisons for any attribute *a* is $|dom(a)|$ and the search space can be as large as $|dom(a_1)|\times\ldots\times|dom(a_n)|$ where *n* is the number of attributes in *R*.

The brute-force solution is to enumerate all possible conjunctive equality selection conditions over dataset *R*, compute the metric function for each conjunctive condition and store DQ of the results and the conjunctive selection condition in DQ profile *Pr*. *Pr* can then be queried to estimate the quality of the result set for any given query with any selection condition.

The brute-force browsing of all possible conjunctive conditions is inefficient for two reasons: First, there are many selection conditions for which no result in the dataset can be found. For example if data set *Item* does not include tuple *{Cannon, SLR, Low}* (or *{C,S,L}*), it is not required to be checked. Second, measurement of the quality of the query result for each combination of conditions requires execution of a query (first, the query result set should be found, and then the data quality of that result can be calculated). Obviously there is a cost associated with each query.

B	M	P	I
C	S	H	1.Jpg
C	S	H	2.Jpg
C	S	L	
C	N	H	4.Jpg
S	S	H	5.Jpg
S	S	H	6.Jpg
S	S	H	
S	S	L	7.jpg
S	N	H	
S	N	L	8.Jpg
S	N	L	

(a)

B	M	P	#	Q
C	S	H	2	2
C	S	L	1	0
C	S	--	3	2
C	N	H	1	0
C	N	--	1	0
C	--	--	4	2
S	S	H	3	2
S	S	L	1	1
S	S	--	4	3
S	N	H	1	0
S	N	L	2	1
S	N	--	3	1
S	--	--	7	4
--	--	--	11	7

(b)

Figure 2 A sample conditional DQ profile's initial generation (a) the source data set (b) the expansion phase of conditional DQ profile

Efficient Conditional DQ Profiling: In this section we propose a technique for creation of the conditional DQ profile and further reduce the size. We call the first phase of the technique as expansion phase where the initial conditional DQ profile is created. We call the second phase as reduction phase in which we reduce the profile size while maintaining acceptable accuracy of DQ estimation. We further have another phase called *revert* that will be used to overcome any loss of information incurred during the reduction phase. The level of acceptability of accuracy is ensured by allowing the user to define two thresholds which will be discussed further in this section.

Let $a_1 \ldots a_k$ be the list of attributes to be used for query conditions over R, and m_{a_i} be the metric function m on attribute a_i to be profiled.
In initial phase, we create conditional DQ profile Pr using the following query:

select $a_1 \ldots a_k$, *count*(a_i) *as #, sum*(m_{a_i}) *as Q from R group by* $a_1 \ldots a_k$
union
select $a_1 \ldots a_{k-1}$, *'-' as* a_k, *count*(a_i) *as #, sum*(m_{a_i}) *as Q from R group by* $a_1 \ldots a_{k-1}$
…
select a_1, *'-' as* a_2, *…, '-' as* a_k, *count*(a_i) *as #, sum*(m_{a_i}) *as Q from R group by* a_1
union
select '-' as a_1, *…, '-' as* a_k, *count*(a_i) *as #, sum*(m_{a_i}) *as Q from R*
order by a_1, *…,* a_k

Above query creates the conditional DQ profile similar to Figure 2 (b). We now explain the above concept in the context of our running example. In Figure 2 we use completeness for metric function defined as one simple rule: "if Image is Null then 0 else 1". Note that the same principle can be applied to other DQ metrics that can be defined through a rule or a set of rules. Figure 2 (a) shows sample data set Items(Brand, Model, Price, Image). All attributes and their values are abbreviated to the first letter in Figure 2. Figure 2 (b) illustrates the conditional DQ profile before applying the ε threshold for estimating the completeness of attribute Image. Profile created after the expansion phase, is usually large, therefore; we take further steps to reduce the size of generated profile.

```
Data: Profile Table Pr

create empty stack S
push(S,fetch x from Pr)
while cof(Pr) do
    Fetch x from Pr
    if x is parent of top(S) or vice versa then
        Let s := top(S)
        if (s.Q/s.#) − ϵ ≤ (x.Q/x.#) ≤ (s.Q/s.#) + ϵ then
            delete x from Pr
    end
    else
        while top(S) is not parent of x or vice versa do
            pop(S)
        end
    end
end
return Pr
```

Algorithm 1 Reducing the size of conditional DQ profile

Approximating DQ to Reduce Size: We reduce the size of the profile in two ways. First by estimating the value of query's DQ metric instead of providing the exact value and second, by removing all the conditions that return few and statistically insignificant number of records. These reductions are controlled by two thresholds: First, minimum set threshold τ which defines the minimum size of tuple set to be profiled. For example if there is only four Panasonic cameras in a shop, and $\tau = 10$, DQ profile row with condition *Brand=Panasonic* will not be created. This threshold will reduce noise in DQ profile, e.g. if this threshold is ignored ($\tau=1$), any single tuple which has the metric function value of exactly 0 or 1 appears in the conditional DQ profile. Second, certainty threshold ε is the maximum acceptable uncertainty for the user. For example, if $\varepsilon=0.1$, the DQ profile row with condition *Brand=Cannon* should return completeness between 40% and 60% for data set in Figure 1.
Algorithm 1 receives conditional DQ profile table *Pr*, and accuracy threshold ε as input, and returns a reduced conditional DQ profile table as output. It removes profile records when their metric function value is within the range of ε from their parent profile record. For a given profile record Pr_{t_1}, a parent profile record Pr_{t_2} is a record which for every attribute value in Pr_{t_1}, has the same value or "-" exists in Pr_{t_2}. For example *{C,S,_,4,2}* is parent of *{C,S,H,2,}*.

Algorithm 1removes profile records from dataset where quality of the profile record is in the range of ε from the quality of the closest parent record that is not deleted. For example as in Figure 3, record *{S,_,_}* for which the quality (57%) is in the range of $\varepsilon = 0.2$ (or 20%) from its parent *{_,_,_}* (63%) is removed from dataset. Hence, the quality of a query with condition *Brand=S* will be estimated from its closest parent 63% which is not far from the actual quality of the result-set (around 57%). Figure 3(a) depicts the DQ profile from Figure 2 (b) considering $\tau=2$ and Figure 3(b) shows the reduced profile table of the Figure 3(a) considering $\varepsilon = 0.2$.
Furthermore, reducing the size of profile by removing profile records that reflect less than τ records is straightforward and it can be done using the command *delete* from *Pr* where $\# \leq \tau$.
Although, Algorithm 1 is able to reduce size of the profile, it has the side effect of losing some valuable data. For example if DQ profile record *{S,S,H}* is removed, the quality of the query with condition *Price=H* cannot be estimated. Using Figure 3(a), quality of *{_,_,H}* can be estimated from the quality of records *{C,S,H}*, *{C,N,H}*, *{S,S,H}*, and *{S,N,H}* (i.e. (2+2+0+0)/(2+3+1+1)=57%), but using Figure 3(b), such value cannot be estimated correctly. This problem appears because we have reduced a record by comparing it to only one of the possible parents. For example *{S,S,H}* is not only a child of *{S,S,_}* in the search space. It can also be a child of *{S,_,H}* or *{_,S,H}* which do not exist in the generated DQ profile.
After reducing the DQ profile with the threshold ε, we reduce the profile with minimum set threshold τ.

B	M	P	#	Q	%
C	S	H	2	2	100
C	S	--	3	2	66
C	--	--	4	2	50
S	S	H	3	2	66
S	S	--	4	3	75
S	N	L	2	1	50
S	N	--	3	1	33
S	--	--	7	4	57
--	--	--	11	7	63

(a)

B	M	P	#	Q	%
C	S	H	2	2	100
S	N	--	3	1	33
--	--	--	11	7	63

(b)

Figure 3 Reduced conditional DQ profile of Figure 2 with threshold τ=2 and ε=0.2.

To resolve this problem, if we are removing some records (nodes in the search space) from the profile, we should revert all possible parents of the removed nodes if removal of the node affects the correctness for DQ estimation for its parent. For example, if we are removing *{S,S,H}*, and the quality of its parent node *{_,S,H}* is different, we should revert *{_,S,H}* into the conditional DQ profile. We revert any useful deleted data as follows; we refer to this part of the algorithm as Revert phase which works as follows.
Given profile table *Pr* $(a_1, ..., a_n, \#, Q)$, and all eliminated tuples $C=(c_0, \ldots, c_n, \#, Q)$, and a prefix list consisting of $A=(a_i, \ldots, a_k)$, let A^+ be the attributes of *Pr* that do not appear in *A*. First; running expansion and reduction phases for *C and the results are* sorted by ordering the sequence a_j where a_j starts with sequence *A* and continues in a random order from A^+. Revert results to *Pr* with the depth of the recursion attached to it and keep reduced records in *C'*. Then, if $|C'| \leq |C|$ or A^+ is *null* or *C* is *null*, each attribute $a' \in A^+$, recursively run revert phase with parameters $A=A \cap a'$, *C'* and *Pr*. For example deleted records from Figure 3(a) will be sent to revert phase in parallel, sorted as *{P,M,B}*, *{M,P,B}*. First recursion for the attributes sorted as *{P,M,B}*, generates *{H,S,S}, 66%, {H,S,_},66%, {H,_,_},66%*, etc. It will be reduced to *{H,_,_},65%*, etc. The revert phase re-runs for another level as long as size of the result of the function is reducing.

Complexity of the Proposed Algorithms

Initial expansion phase of the algorithm, consists of *n* group by queries, where *n* is the number of attributes that may appear in conditions. Group by queries run in $O(|R|)$ where $|R|$ is the size of the relation *R*. Since pushes and pops to the stack in the Algorithm 1 happen exclusively, they do not introduce extra complexity dimension. Complexity of the expansion and reduction phases of the algorithm is $O(|R|.n)$ in the worst case. For the Revert phase of the algorithm we conduct various

experiments to compare execution time versus dataset size.
In traditional DQ profiling, a full scan of database is required to figure out the average DQ profile of the whole dataset per metric $O(|R|.n)$. However, expansion phase of the algorithm usually generates a much larger DQ profile. Reduction and revert phases of the algorithm take more preparation time in favour of a smaller conditional DQ profile.

4. QUERYING CONDITIONAL DQ PROFILE TO ESTIMATE QUERY RESULT

Conditional DQ profiles can estimate the quality of the result set limited to the two factors τ and ε. There might be cases where query conditions do not exist in the profile. This may happen for three reasons: 1) If the query returns no data. We assume checking for this situation occurs before querying the profile to avoid unnecessary reference to datasets that are not relevant. 2) The profile record might have been removed due to ε threshold. In this case, we return estimation for next available parent of the query in the profile. 3) The profile record might have been removed due to threshold τ. In this case we act similarly to the previous case, but the result can be inaccurate.
In previous section we proposed techniques to create conditional DQ profile for a given data set. Estimating DQ of the query result from a conditional DQ profile is possible by finding the first row in the conditional DQ profile that contains all conjunctive selection conditions in the query. In the worst case, a full scan of DQ profile may be required. Hence speed of query answering from conditional DQ profile is proportional to the size of DQ profile.
Consider relation *R* from the source *S*, and conditional DQ profile *Pr* for an arbitrary metric and attribute. *Pr* consists of columns (a_0, …, a_k, #, Q). To estimate the quality of the result set for a user query consisting of selection criteria $\varphi_{a_0=\alpha_0 \wedge \ldots \wedge a_j=\alpha_j}$, same query should be run against the DQ profile *Pr* (instead of *R*), and it should also consider the don't care values "_". We can translate φ to $\varphi'_{a_0=\alpha_0 \wedge a_1="\_",\ldots}$. If a_i is not mentioned in the query conditions, it still appears in the translation as a_i="_".
For example query *SELECT * FROM D WHERE Brand= "Cannon" AND Model= "SLR"* translates to *SELECT TOP 1 #, Q FROM Pr WHERE (Brand= "Cannon" OR Brand= "_") AND (Model= "SLR" OR Model = "_") AND (Price="_") ORDER BY Brand, Model, Price*. In regards to the ORDER BY operation, don't care "_" value should appear after any other value in the domain, hence, general result will be selected only if a less general result does not exist.

Scalability to Very Large Databases

Creating a Conditional DQ profile using the proposed algorithms for very large databases can become expensive, and the profile itself can also become very large. In many applications it is impractical to run such expensive queries for each available data source. Our suggestion for such scenarios is that the query engine should start with empty profiles, and monitor caches for all query results. It then generates Conditional DQ Profile rows from query result caches. Thus, profile rows will be completed gradually, and only parts of a dataset that are queried by users will be profiled. Downside is that no estimation can be made for new queries if their result is not a subset of an already profiled query. Cached based generation of conditional DQ profiles is part of our future work.

5. EVALUATION

In this section we study effectiveness and scalability of our proposed techniques. Effectiveness of conditional DQ profile is compared against the traditional DQ profile. Traditional DQ profile generates only one value for the whole data set regardless of the underlying data. Therefore, it imposes very low overhead on the system. When distribution of dirty data is totally uniform through the whole dataset, our conditional DQ profile also seems to be very small. However, the benefit of our technique becomes obvious when distribution of dirty data in the data set is skewed (i.e., non-uniform), which is the norm for all realistic workloads.

In this section we only compare the effectiveness of our proposed approach with traditional DQ profiling. With regard to scalability of profile creation, traditional DQ profile has a constant cost, which is not comparable to our technique. However, we do study the scalability of our technique and its sensitivity to various factors. The presented experiments are based on the publicly available DBLP data set [4].
In this paper, we assumed that details for calculating the DQ metric are transparent to our technique. Hence we write a deterministic function that pre-generates a DQ metric result value for every record of the dataset. This function stores the pre-generated results per record in a lookup table and returns the same result for every record of the dataset each time.
Meanwhile, the distribution of dirty data within the data set is one important parameter in our experiments since it affects the size of our proposed Conditional DQ Profile.
On one end, if the distribution of dirty data is totally uniform then our conditional DQ profile will be reduced to a traditional DQ profile and would need only one row to satisfy the DQ estimation requirements of all the queries. On the other end, if distribution of dirty data for every subset of the dataset is independent from distribution of dirty data from the rest of the dataset, conditional DQ profile should have one profile record for every record of the database.
In order to simulate different distributions of dirty data in a data set, we pre-generate results for a hypothetical metric function. In Definition 1, we defined metric function to be a function that returns a Boolean value for every given record from data set. We simulate a metric function with a lookup table that assigns 0 or 1 to every record of the data set and we simulate various distributions of dirty data for the metric function (lookup table) as follows: Let *d* be the variation in distribution of dirty data. *d=0* is no variation in distribution of dirty data, i.e. uniform distribution of dirty data, and *d=1* is the maximum variation in distribution of dirty data. We approximately simulate *d* by grouping the dataset by all attributes that can be used in the conditions. Then we generate *count(Groups)/d*, (or *1* if *d=0*) random numbers and map these numbers to groups with the same pattern. For example, there might be *100* different groups for *{Brand, Model, Price}*. If *d=0.5*, two random numbers, e.g. *0.3* and *0.9*, will be generated and mapped to the groups to introduce dirty data respectively. Therefore; division of dirty data for first *50* groups will be *0.3* and division of dirty data for the other *50* groups will be *0.9*.
To evaluate our technique, we first generate the conditional DQ profile. To ensure we have covered different combinations, in each experiment we run *10%* of all possible queries with conjunctive equality selection conditions against the profile. In order to better understand the interplay between the two different thresholds (i.e., ε and τ) on DQ estimation quality, in the experiments where we measure the impact of ε, we eliminate any query that returns less than τ results from the generated workload.

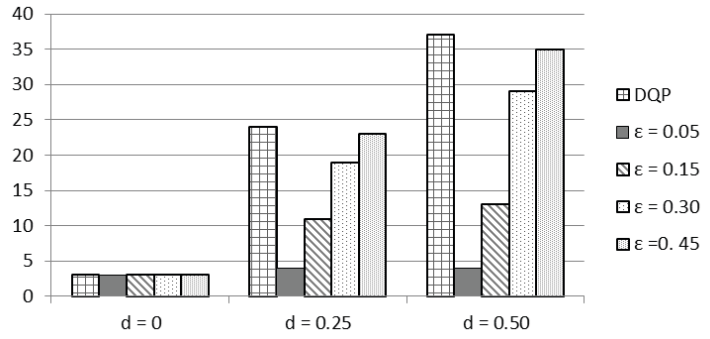

Figure 4 Comparison of the average estimation error for traditional DQ profile (DQP) and conditional DQ profile with different certainty thresholds (ε=0.05 to ε=0.45) and for different variations in distribution of dirty data *d*.

Effectiveness of DQ Estimation: Figure 4 shows the average estimation errors of randomly generated queries. Figure 4 compares the average estimation error rate for both traditional DQ profiles (DQP) and

conditional DQ profiles (CDQP) when varying the distribution of dirty data. For each query, we computed the Estimation error as the difference between the actual and the estimated values of the DQ metric function of that query result. Conditional DQ profile is generated for different thresholds $\varepsilon=0.05$, $\varepsilon=0.15$, $\varepsilon=0.30$, and $\varepsilon=0.45$. It can be observed that the estimation accuracy using our conditional DQ profile (i.e., CDQP) is always less than ε. That is, DQ estimation error rate in conditional DQ profile will not exceed the threshold ε.

When $d=0$ (i.e., the distribution of dirty data is totally uniform), traditional DQ profile performs as well as conditional DQ profile. However, when the variation in the distribution of dirty data increases, more queries will experience a deviation between the value of the actual metric function and that estimated by the traditional DQ profile leading to a significant increase in estimation errors (as shown in figure).

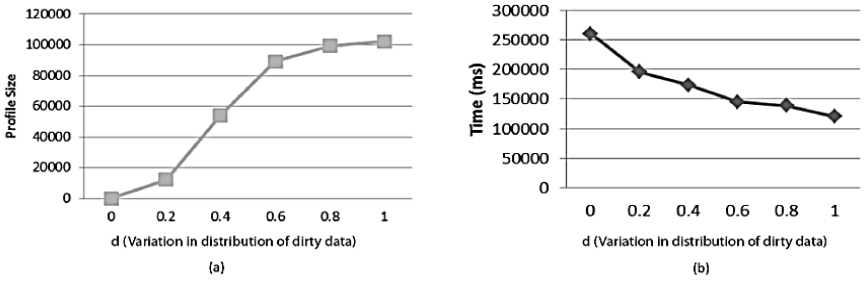

Figure 5 (a) Effect of variation in distribution of dirty data *d* on profile size (b) Effect of error distribution on profile generation time.

Effect of Error Distribution: Figure 5 (a) and (b) show the effect of variation in distribution of dirty data on the profile size and the cost of creating conditional DQ profile. We conducted this experiment on a Windows 7, 32 bit virtual machine with 4GB of RAM on an Intel Core i3 2GHz machine running OSX as the host operating system.

It can be observed that uniform distribution of dirty data results in the minimum profile size. Indeed, if for example from every 10 records in the dataset, 7 are of good quality, any arbitrary subset of the dataset will return 70% for the metric function. Hence, it might be enough to only keep the metric function result for the whole data set. However, when the skewness of dirty data increases, a bigger profile will be required.

Based on Figure 5 (b); although the profile size increases with increasing the skewness of dirty data distribution, profile generation time decreases. The reason for this behaviour is that most of the profile generation time is spent in the reduction phase. When dirty data distribution is more skewed, fewer records are removed from the profile generated in the expansion phase, hence, fewer amount of data is sent to the reduction phase.

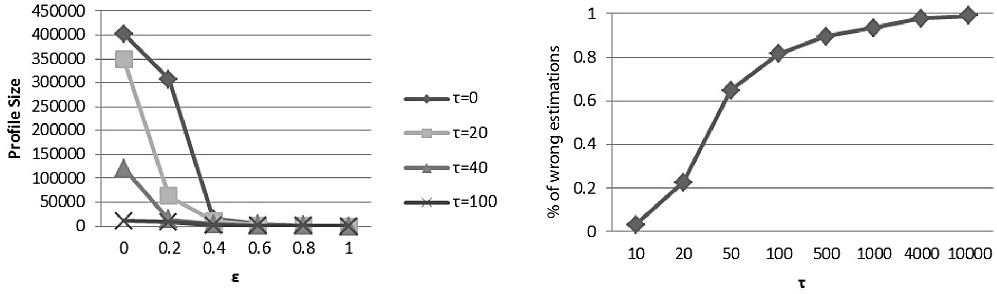

Figure 6 (a) Effect of certainty threshold ε on the size of DQ profile for different minimum-set thresholds τ (b) Effect of τ on percent of correct estimation made using the DQ profile.

Effect of ε and τ Thresholds: Figure 6 studies the effect of both minimum set threshold τ and accuracy

threshold ε on size of the conditional DQ profile. It can be observed that with tolerating a degree of uncertainty (as defined by the certainty threshold - ε), a significant reduction in the DQ profile size can be achieved. Thresholds τ and ε improve size reduction if they are used together, e.g. a very low τ and relatively high ε will result in a very large profile size. Figure 6 (b) depicts the effect of minimum set threshold τ on the percentage of the correct estimations made using DQ profile. It means that DQ of queries which return less than τ records are not estimated correctly, however, if the query returns more than τ records, it is guaranteed that the estimation error will be less than ε. Figure 6 (b) shows that percentage of queries that return less than τ records increase quickly with only small increase in τ. Since DQ is a statistical characteristic of data, $\tau=0$ means that if a query results in only one or two records, DQ profile row for supporting that query should be kept in the profile table, even though, in many applications only a few records, do not convey statistical significance. On the contrary, increasing τ to a big number will remove many possibly valuable profile records from the profile after which the DQ metric for them cannot be estimated, and we may need to fall back to traditional DQ metric estimate for those queries.

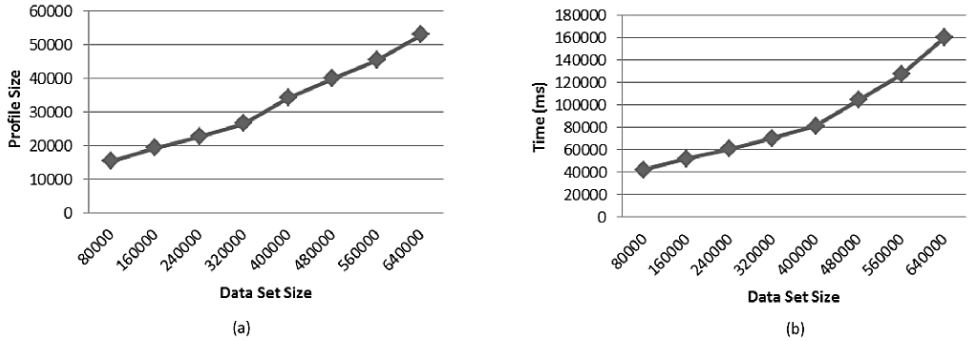

Figure 7 Scalability graphs (a) generated profile size versus number of records in dataset (b) profile generation time versus database size.

Effect of Input Data Size: Figure 7 illustrates the scalability of the proposed algorithms. In this experiment, we initially used a smaller subset of the DBLP dataset and gradually increased the size of that subset until the whole data set is covered. For each experiment we create a new set of queries. Figure 7 illustrates the effect of data set size on the size of the conditional DQ profile. It also illustrates the effect of data set size on the profile generation time. Experiments have been conducted with thresholds $\varepsilon=0.2$ and $\tau=20$ which gives us a good balance between the estimation error and profile size based on Figure 6.
In summary, our experiments show that conditional DQ profile can be used in the query systems to improve user experience with higher quality data without introducing very expensive overheads.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we investigated how meaningful statistical representations of data quality, namely DQ profiles, can be generated in a way that quality of a query result can be effectively and efficiently estimated from it without actually querying data sources. The generated DQ profile should be markedly small compared to the original data set and estimation of the quality of query results from the profile needs to meet accuracy and efficiency requirements. We proposed the notion of conditional DQ profiling to address the above problem, together with related algorithms to efficiently generate conditional DQ profiles and reduce their size where possible. We further note that once the profile is generated, it should be kept in-sync with data source to maintain effectiveness. Update processes for DQ profiles can be periodic or incremental, or can be learnt from the query responses cached in the system. Proposition of an effective method to keep conditional DQ profiles up to date is part of our future works. We also plan to investigate further improvements of our algorithms for conditional DQ profile creation in order to improve the accuracy of DQ estimation while maintaining efficiency thresholds.

REFERENCES

[1] C. Batini and M. Scannapieco. Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications). Springer-Verlag, NJ, USA, 2006.

[2] O. Benjelloun, H. Garcia-Molina, Q. Su, and J. Widom. Swoosh: A generic approach to entity resolution. VLDB Journal, 18(1):255-276, 2008.

[3] P. Bohannon, F. Wenfei, F. Greets, J. Xibei, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. ICDE, 2007.

[4] dblp. http://kdl.cs.umass.edu/data/dblp/dblp-info.html.

[5] T. Friedman and A. Bitterer. Magic Quadrant for Data Quality Tools. Gartner Group, 2006.

[6] L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate String Joins in a Database (Almost) for Free. VLDB, 2001.

[7] L. Gravano, P.G. Ipeirotis, N. Koudas, and D. Srivastava. Text Joins for Data Cleansing and Integration in an RDBMS. ICDE, 2003.

[8] L.V.S. Lakshmanan, N. Leone, R. Ross, and VS Subrahmanian. ProbView: a flexible probabilistic database system. ACM Transactions on Database Systems, (TODS), 22(3):419-469, 1997.

[9] F. Naumann. Quality-Driven Query Answering for Integrated Information Systems. LNCS 2261, 2002.

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

[11] Y.L. Simmhan, B. Plale, and D. Gannon. A Survey of Data Provenance in e-Science. SIGMOD RECORD, 34(3):31, 2005.

[12] R.Y. Wang, V.C. Storey, and C.P. Firth. A framework for analysis of data quality research. TKDE, 7(4):623-640, 1995.

[13] N. K. Yeganeh, S. Sadiq, K. Deng, and X. Zhou. Data Quality Aware Queries in Collaborative Information Systems. APWeb, 2009.

[14] N.K. Yeganeh and S. Sadiq. Avoiding Inconsistency in User Preferences for Data Quality Aware Queries. In BIS, 2010.