Formal Approach to Data Accuracy Evaluation

Belkacem Athamena[1] and Zina Houhamdi[2]
E-mail: athamena@gmail.com, belkacem.athamena@aau.ac.ae, z_houhamdi@yahoo.fr, zina.houhamdi@aau.ac.ae
[1]Business Administration Department, College of Business, Al Ain University, United Arab Emirates
[2]Cybersecurity Department, College of Engineering, Al Ain University, United Arab Emirates

Keywords: data integration systems, data quality, data accuracy

Received: December 20, 2019

Usually, data quality is defined by multiple attributes that allow classifying the output data (such as completeness, freshness, and accuracy) or the methods exploiting these data (such as dependability, performance, and protection). Among the suggested quality attributes, we will discuss one of the principal categories: data accuracy. Scientific experiments, decision–making, and data retrieval are examples of situations that require a formal evaluation approach to data accuracy. The evaluation approach should be adaptable to distinct understandings of data accuracy and distinct end-user expectations. This study investigates data accuracy and defines dimensions and metrics that affect its evaluation. The investigation of data accuracy generates problems in the user expectation specification and database quality models. This work describes our proposed approach for data accuracy evaluation by defining an evaluation algorithm that considers the distribution of inaccuracies in database relations. The approach decomposes the query output in accordance with data accuracy, labels every part with its accuracy value, and addresses the possibility of enforcing data accuracy by using these values. This study mainly contributes by proposing an explicit evaluation of quality attributes of data accuracy, a formal evaluation approach to data accuracy, and suggesting some improvement actions to reinforce data accuracy.

Povzetek: Opisana je formalna metoda preverjanja točnosti podatkov.

1 Introduction

Data quality has increasingly become an essential characteristic required by users particularly for data integration systems (DISs), which involve combining data residing in multiple databases and providing the user with a unified view of these data as answers to their queries [4]. Because of the growth of retrieved data, users are becoming increasingly worried about data quality. On the other hand, the number of quality attributes and their relationships are huge. Accordingly, data quality evaluation is considered as a complex problem involving multiple variables. In DIS context, data quality evaluation is exceptionally complicated because of the combination of data derived from different databases with possibly distinct qualities. Because of the large number and high heterogeneity of databases that are independent, it is crucial to precisely determine their quality and to consider it in the design phase of DISs.

System quality improvement is associated with optimization of a problem with multiple variables, which may be very complex specifically in an indefinite context [3, 26]. Consequently, it is arduous to consider all quality attributes at the same time. To investigate data quality thoroughly, it is inevitable to investigate each quality attribute independently besides the factors of the context that affect it. Dependencies will be investigated later. Among the proposed quality attributes, we opt for the main category: data accuracy. Currently, many systems consider the need to have reliable measurements of data accuracy as a crucial and decisive requirement. These systems are numerous and are used in diverse fields, such as customer relationship management, web-services integration, scientific experiments, decision–making, and data retrieval.

This study discusses data accuracy analysis in DIS and considers the relational scenario. Explicitly, it treats user queries that are composed of projections, selections, and joins (PSJ) operators over a collection of database relations. It addresses the problem of accuracy evaluation of data delivered to the user in response to user queries and decides if the expectations of the user on the data accuracy can be achieved or not.

Our approach consists of fragmenting the query output in sets of tuples (called areas), which have uniform accuracy values, and marking each area with its accuracy value. Consequently, the user can do the following:

- extract only the most precise data by selecting the area that possesses high accuracy values,
- exclude data that do not satisfy the accuracy threshold by ignoring areas possessing low accuracy values, or
- classify data by sorting areas based on their accuracy values. Moreover, if we desire displaying additional data (because the first accurate area is incomplete), we can show the next area and so on. Thus, this will represent an added value to the delivered data.

This paper illustrates our proposed accuracy evaluation algorithm. We present the values of semantic accuracy using the Boolean metric where each cell of every area of any relation is assigned a value accurate or inaccurate. However, the accuracy of the whole area is calculated as an aggregation of the cells' accuracy values by dividing the number of accurate cells by the total number of cells in the area. The query results are sorted by the areas' accuracy values. All areas with accuracy value bigger than the user threshold are considered as area with high accuracy value and on the other hand, the areas with accuracy value less than the user threshold are considered as area with low accuracy value.

The fragmentation of query output and evaluation of data accuracy requires an in–depth analysis of the inaccuracy distribution in database relations and their union to generate the query output [24]. For this purpose, we split database relations into areas possessing uniform accuracies. As clarified in [15], databases are usually of heterogeneous quality; consequently, assigning an accuracy value for the entire relation is considered as an imprecise accuracy calculation of particular data [14]. The area is described as a view (selection and projection) over the relations of the database. That is to say, a fragment aggregates a group of relations specified by the predicates characterizing the fragment [8, 9, 25].

This paper presents a formal approach to evaluate data accuracy that considers the portions of database relations and processes them to generate an output for user query. It focuses on pre–evaluation, that is, the data accuracy is estimated before the execution of the user query. The outputs of our evaluation approach will be useful to make a comparison between different query plans to select the plan with the maximal accuracy value. Furthermore, these outputs are beneficial during the design phase (to determine which databases will be included in the DIS) and monitoring phase (to calculate the query accuracy). Eventually, this study discusses the issue of accuracy improvement. It proposes the usage of output fragments to choose the areas with high accuracy values. We use the portions of a relation having the highest accuracy value instead of using the whole relation. This distinguishes our approach from existing approaches in the literature.

2 Background

Data accuracy symbolizes a set of quality attributes. This section describes the three accuracy attributes listed in the literature [23, 27]:

– *Syntactic accuracy* represents the level at which data do not contain syntactic mistakes such as format inconsistencies and spelling errors. It expresses the interval between descriptions of the data in the DIS and conventional descriptions of these data (syntactic gap).

– *Semantic accuracy* defines how adequately the data describe the environment state. It represents the interval between the data described in the DIS and real–world data (semantic gap).

– *Precision* describes the level of data details. It expresses the gap between the detail level of the DIS data and its planned detail level.

Apropos of accuracy measurements, three metrics are mentioned in the literature [12]:

– *Boolean metric* uses a Boolean value to indicate if the data detail is accurate (1 or True) or inaccurate (0 or False).

– *Degree metric* uses a dimension to capture the principle of how precise the data are; this usually belongs to a $[0 - 1]$ interval.

– *Value–deviation* is defined as an integer to capture the gap between data item in the system and the original one; this is usually normalized to a $[0 - 1]$ interval.

This section reviews the main concepts used in this study. First, it discusses current approaches for evaluating data accuracy as adapted from our proposed model, particularly, the fragmentation technique, and it recalls the characteristics of partitioning relational databases. It reviews the concept of query rewriting and explains the *bucket* algorithm. Finally, it comments on selectivity estimation techniques.

Our accuracy evaluation approach is supported by two techniques: *prior evaluation* approach, which presumes homogenous distribution of errors [16] and *posterior evaluation*, which uses the accuracy homogeneity for partitioning database relations [19].

Prior Evaluation Method: Naumann et al. [16] proposed a method that propagates quality factors (encompassing data accuracy) in conformity with query operators. The method estimates the query output accuracy based on the accuracy of the database. The database relation accuracy is the percentage of syntactic accuracy (rate of accurate cells).

The query is a PSJ. The method assumes that inaccuracies are homogeneously spread in the database relations; thus, regardless of the selected tuples or projected attributes, the database accuracy is conserved. Concerning the join operation, the joined data accuracy is computed by multiplying the accuracy of input relations [20].

The disadvantage of this method is the strong assumption on a homogenous error distribution (usually inapplicable to real–world data). Generally, query operations do not maintain accuracies and accordingly, this method does not obtain an exact calculation of the accuracy. This deficiency is due to the absence of knowledge on inaccuracy distribution (where the errors are concentrated). Supplementary knowledge characterizing instances of relation is mandatory to obtain outputs that are more accurate.

Posterior Evaluation Method: This method uses a partitioning algorithm [15, 19] for fragmenting the database relations into areas that have extremely uniform accuracy. An area is described as a view, which involves projection and selection operations. The accuracy is calculated by considering the portion of each relation and then computing the cell accuracy of that portion. The area accuracy is the ratio of semantic accuracy (rate of correct cells). The accuracy values will be exploited in portion fragmentation by applying a computerized algorithm for fragmentation that evaluates a set of criteria. After that, the same fragmentation is performed over the complete database relation.

Usually, the query is a PSJ and relational algebra is employed to operate with the partitions, i.e., the operator takes as inputs the set of relations with their respective partitions. Its output is a single relation with its partitions: the partition of projection (selection) is determined as the intersection of the operation output and the partition of the input relation (intersection of selection conditions with projected attributes). In this case, the data accuracy is conserved owing to accuracy uniformity. Finally, when the query output is ready, we calculate the accuracy value as the weighted sum of area accuracy (weight is equal to the number of cells in the area). Note that a single value of accuracy is computed for the entire output.

Partition Algorithm: We will concisely describe Rakov's algorithm for sampling a relation [19]. Figure 1 shows the related pseudo–code. It is an iterative function using a classification tree. It takes a relation as input and splits it into two blocks (vertical or horizontal splitting but never both) and it tries to identify the block with the highest uniformity; then it iterates the procedure for each block.
The block partitioning terminates if it provides a negligible amelioration in uniformity. A threshold x notes a fair uniform distribution of accuracies in the block and it serves as a stopping condition. Uniformity is estimated using Gini indices [6]. The Gini index $GI(P)$ formula is given by equation 1:

$$GI(P) = 2r(1-r) \tag{1}$$

where r represents the rate of accurate cells in partition P. The equation 2 defines the halt constraint that estimates the reduction of the partition of the Gini index:

$$\Delta GI = GI(P) - \alpha_1 GI(P_1) - \alpha_2 GI(P_2) \tag{2}$$

where $\{P_1, P_2\}$ are partitions of P and $\alpha_i = \frac{|P_i|}{|P|}, i = 1, 2$.

Because consideration of the complete possible relation partitions is excessively costly, Rakov suggests some heuristics to decrease the number of treated partitions. In addition, we distinguish two types of attributes: categorical and ordered [1]. In the case of horizontal partitioning, if the attribute is ordered and possesses K different values $(A_1 \leq \cdots \leq A_k)$, the partitions are identified as $(K-1)$ binary conditions $t \leq A_i$. Otherwise, in the case of a categorical attribute that possesses K different values, these

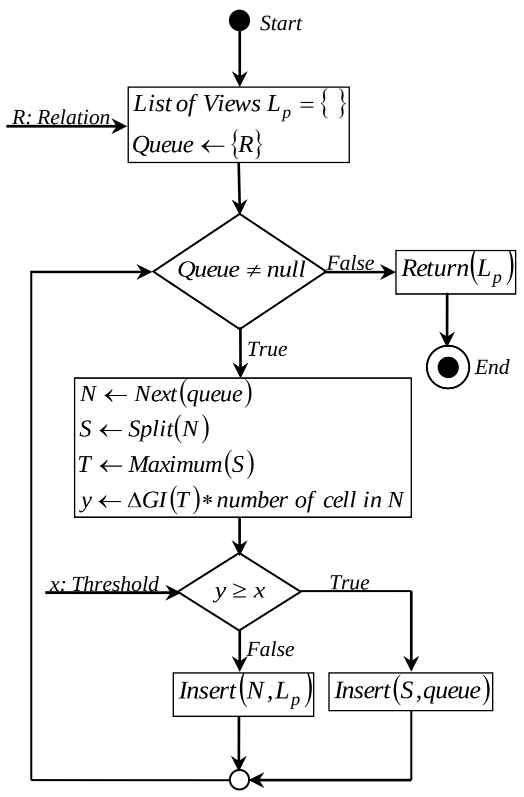


Figure 1: Relation Partitioning Algorithm.

values are sorted based on the number of incorrect cells. After that, they are considered as ordered attributes. For vertical partitioning, we consider all partitions for a small number of attributes; otherwise, for a large number of attributes, we apply the same method employed for categorical attributes. In the following, we discuss the characteristics of well–constructed partitions.

Partition correctness : There are three correctness constraints that a partition must satisfy to guarantee database coherence [17]. The constraints are related to the following:

- *Disjunction*: The horizontal decomposition of a relation R into partitions $R_1, \ldots, R_n$, verifies that $\forall cell\ d_i / d_i \in R_j \Rightarrow d_i \notin R_k, k \neq j$ (i.e., each cell in the partition R_j does not appear in any other partitions R_k where $k \neq j$). This property ensures that the partitions are disjoints. This rule guarantees that there is no intersection between all horizontal partitions. In the case of vertical decomposition, the disjunction is limited to non-primary key attributes of the relation because the primary key attributes are basically duplicated in all partitions.

- *Restoration*: In the case of decomposing a relation R into partitions $R_1, \ldots, R_n$, there is always a way to find an operator $\Omega/R = \Omega R_i$, $i = 1 \dots n$. This rule assures the preservation of data restrictions defined as dependencies. Normally, the union operator is used for horizontal partitioning whereas the join operator is used for vertical partitioning.

- *Completeness*: Assuming that the decomposition of a relation R into partitions $R_1, \dots, R_n$, all items belonging to relation R also belong to one or more R_i partitions. This characteristic ensures data preservation, i.e., there is no data loss (all data in global relation are projected in partitions). For horizontal partitioning, items denote tuples; however, for vertical partitioning, they denote attributes.

Query rewriting: is the reformulation of a user query (defined by the global relation) to a possible analogous scheme, known as *rewriting*, which concerns exclusively the database framework [11]. In the local–as–viewed (LAV) strategy, the global relation is defined without referring to the databases, and after that, a mapping between them is made by expressing each database relation as a view over the global relation [21]. Thus, query rewriting is comparable to the application of views to answer a user query.

Definition: For a query Q over relations R_i defining the global relation and views V_i referring to database relations over R_i, the query Q_r is called a rewriting of Q if:

- $Q_r \subset Q$
- Q_r concerns exclusively the views.

In general, the query is a PSJ and is defined using a datalog–like language. Thus, query Q is expressed by equation 3:

$$Q(X) = R_1(Z_1) \wedge \cdots \wedge R_n(Z_n) \wedge C_Q \qquad (3)$$

where

- R_i *is a relation* and $Z_i = \{A_i\}/A_i$ *is an attribute* of R_i
- C_Q is a *conjunction* predicate, $C_Q = u\theta v/\theta \in \{=, <, >, \leq, \geq\}$, and $u, v \in \bigcup_{1\leq i\leq n} Z_i$
- $X \subseteq \bigcup_{1\leq i\leq n} Z_i = \{A_i\}/A_i$ projected by Q

Numerous algorithms for query rewriting can be found in the literature and [7] presents a review of these algorithms. The most popular is the bucket algorithm [13], which we will describe and use in this study.

The bucket algorithm computes all possible rewritings that are included in (but not imperatively analogous to) the initial query [5]. The algorithm prunes the area of candidate rewritings in two phases:

- $\forall t \in Q$: Construct a container called bucket that includes the contributing database relations for t, i.e., the database relations containing tuples of t.
- Construct candidate rewritings (query integration by joining one database relation from each bucket) and preserve only the rewritings belonging to Q.

The first phase possesses a polynomial complexity with respect to the database number. However, the second phase reduces the complexity considerably by diminishing the number of possibilities. Despite the fact that containment is generally manageable, its resolution is equal to the query size (usually small) and happens when the query has many occurrences of exact schemas; accordingly, the containment complexity is not a challenge in real applications [5].

Selectivity Evaluation: Techniques for selectivity evaluation are widely practiced in query optimization. The statistics of data saved in the database provides an estimation to the optimizer [22]. The most popular statistics are histograms, which are adopted by several business database systems. This section shows their current application to evaluate the selectivity of a complex query.

The histogram of attribute A is defined as a list of buckets, where bucket b_i defines r_i, which is a subarea of A's area, and possesses two attributes:

- periodicity p_i, which represents the number of tuples x verifying $x.A \in r_i$, and
- discrete value dv_i, which represents the number of different values of $x.A$ in all tuples $x/x.A \in r_i$.

The query selectivity is calculated by dividing the query cardinality by the relation cardinality. In order to evaluate the query cardinality, we sum the periodicity of all buckets included (totally or partially) in the predicate. If the query has many extended predicates, the selectivity is calculated as the product of all selectivities.

For a random PSJ query, there is a new challenge: cardinality evaluation needs statistical information propagation over predicates, i.e., we must create histograms for in–between outputs using the histograms of database relations. The propagation of in–between statistics in addition to multiple query operators can considerably decrease the accuracy. A possible solution is a precalculation of statistics for a subset of query results that are exceptional and to use them in selectivity calculation of in–between outputs.

The remainder of this paper describes the proposed model for evaluation of data accuracy. Algorithms and methods reviewed previously are modified and adopted in data accuracy estimation.

3 Formal model

This section formalizes the proposed approach for accuracy assessment. The approach considers DIS in relational context (the system combines data from relational databases and allows users to execute queries over a global schema). Each query is reformulated according to the database relations in order to obtain a set of rewritings that select data to answer the query.

The accuracy of data delivered to a user as answer to the user's query is addressed in this section. We arrange data in areas with uniform accuracy to apprise the user on the inaccuracy distribution. To achieve this goal, database relations are decomposed into partitions (called areas) possessing uniform accuracy. Note that areas are views (virtual relations) determined by the distribution rules (projection attributes and selection conditions). The user query is reformulated over the areas (rather than in the global relation). The proposed approach uses an *a priori* assessment method: before performing the query rewriting, we assess the data accuracy, depending only on the operations (PSJ) that define the rewritings and the area accuracy. We can state the problem as follows:

- *Input:*
  - User query
  - A collection of database relations decomposed into areas having uniform accuracy
- *Output:*
  - A rewriting set answering the user query
  - Accuracy value of rewritings
  - Accuracy value of the query answer

Our proposed scenario for accuracy estimation contains three phases:

- *Database relation fragmentation based on accuracy uniformity*: This stage estimates the accuracy of a portion of individual database relation and uses the estimation results for relation fragmentation. The fragmentation phase is performed at the initial stage (DIS development or regularly) but is isolated from the query appraisal stage.
- *Query rewriting with respect to fragments*: Every query is reformulated using the areas of the database relations. This stage generates a set of rewritings using the areas. The output of the query is the union of the created rewritings.
- *Data accuracy estimation of query outputs*: This stage estimates the accuracy of data generated by the rewritings, using the area accuracy, and aggregates them to calculate the accuracy value for the whole user query output.

To reinforce the data accuracy, a simple adjustment consists of rejecting rewritings or areas with small accuracy values. The rejection can be done at the rewriting generation time (early stage). Thus, our approach can be considered as selective. The next subsections provide the details of each phase.

3.1 Database relation fragmentation based on accuracy uniformity

Using the partitioning approach suggested by Rakov [19], the data accuracy is used for fragmenting each database relation. The purpose of fragmentation is the manipulation of pieces of database relation that are uniformly accurate; in other words, if we try to fragment the accuracy value again, it will approximately stay unchangeable [2]. This subsection defines the fragmentation process and discusses how to extract appropriate fragments.

To simplify the partition usage, the approach suggests decomposing the relation into areas using a horizontal partition (selection predicates) and after that, it decomposes each area into subareas using a vertical partition (subsets of attributes). Fragments should verify the three completeness constraints discussed previously: each tuple exists in a unique area, key attributes appear in all subareas, and non–key attributes appear in a single subarea. To manipulate all attributes (key and non–key) in a similar way (projection of the attributes in unique subarea), the key attributes are duplicated. Consequently, a new key for the subarea (composite key) is generated. This approach is called key expansion.

Key expansion definition: If $R(A_1, \ldots, A_m, \ldots, A_n)$ is a relation, where $m \leq n$, and $(A_1, \ldots, A_m)$ form the key of R, $\bar{R}$ denotes the key expansion of R generated by duplicating the key attributes: $\bar{R}(K_1, \ldots, K_m, A_1, \ldots, A_m, \ldots, A_n)$. The duplicated attributes $(K_1, \ldots, K_m)$ are the expanded keys of $\bar{R}$.
In vertical fragmentation, we project the expanded key in all subareas, and we project all attributes of the initial relation in a unique subarea. The fragmentation process (area and subarea) can be formalized as follows:

Horizontal fragmentation: A horizontal fragmentation of a relation R, represented by $H_{P_1,\ldots,P_m}(R)$ (see equation 4), consists of defining a set of subrelations $\{R_1, \ldots, R_m\}$, named areas created by the application of predicates $P_1, \ldots, P_m$ to $\bar{R}$, where a conjunctive predicate set $\bar{P} = \{P_1, \ldots, P_m\}$ over R, disjoint and complete (each tuple of R verifies a unique P_i):

$$\begin{aligned} H_{P_1,\ldots,P_m}(R) &= R_1, \ldots R_m \\ &= \sigma_{P_1}(\bar{R}), \ldots, \sigma_{P_m}(\bar{R}) \end{aligned} \quad (4)$$

The area is denoted by $< Name, Predicate, N, Key_{Accuracy} >$:

- *Name* distinguishes areas within the relation,
- *Predicate* is the conjunction that determines the area,
- *N* represents the number of tuples in the area (verify the *Predicate*), and
- $Key_{Accuracy}$ is the accuracy value of the key attributes of R.

Vertical fragmentation: For n attribute subsets of a relation R, where $\frac{R(S_1,\ldots,S_n)}{\bigcap_i^n S_i} = \phi$: the *vertical fragmentation* of an area R_i, $R_i \in R$, is expressed as $V_{S_1\ldots S_m}(R_i)$ (see equation 5), which defines a list of views $\{V_{i1},\ldots,V_{in}\}$, named subareas generated by projecting the attribute subsets to R_i. The subareas are disjointed (each attribute of R belongs to one subarea). However, the expanded key K of $\bar{R}$ belongs to all subareas:

$$\begin{aligned} V_{S_1\ldots S_n}(R_i) =& V_{i1},\ldots,V_{in} \\ =& \pi_{k,S_1}(R_i),\ldots,\pi_{k,S_n}(R_i) \end{aligned} \quad (5)$$

The subarea is defined as a three–tuple $< Name, Attributes, Accuracy >$, where *Name* distinguishes the subarea within the relation, *Attributes* define the list of subarea attributes, and finally, *Accuracy* denotes the calculated subarea accuracy.

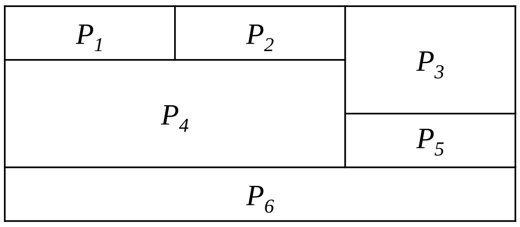
Again, subareas and areas are views determined by fragmentation rules (projection attributes and selection conditions); in other words, the database relations are not really partitioned and saved as independent partitions. Database relations are preserved unvaried in databases and the description models of subareas and areas are kept in the DIS.

After the horizontal and vertical fragmentation of the relation, any relation cell appears in a single subarea belonging to the single area. To fragment database relations, Rakov's algorithm described earlier can be applied. However, it should be noted that the approach alternately executes horizontal and vertical partitioning. Accordingly, the resulting fragments can be different from the areas and their subareas. As a possible solution, we propose to reorganize the fragment in accordance with the horizontal and vertical fragments.

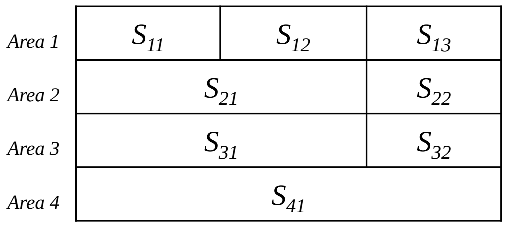
For any hybrid fragmentation (horizontal and vertical) of a relation R, which consists of a set of areas that verify the correctness criteria, it is always possible to obtain different fragments of R by additional fragmentation of some of the areas, i.e., the algorithm obtains the subareas $\{S_{11},\ldots,S_{1m_1},\ldots,S_{n1},\ldots S_{nm_n}\}$, where the subset $\{S_{i1},\ldots,S_{1i_1}\}$ represents the vertical fragmentation of some area A_i ($1 \le i \le n$), and $\{A_1,\ldots,A_n\}$ set represents the horizontal fragmentation of R. To this end, we follow a process that contains four steps:

- Find the selection predicates $P = \{p_1,\ldots,p_r\}$ defining the partition $T_1,\ldots,T_k/r \le k$ (because multiple partitions possess identical selection predicates).
- Search for two non-disjunctive predicates p_i and p_j (i.e., $p_i \bigcap p_j \neq \phi$) and then replace p_i and p_j by $p_i - p_j$, $p_j - p_i$, and $p_i \bigcap p_j$. This step will terminate because the predicates are the unions of inequalities on R's attributes. The output is a set of predicates $P' = \{p'_1,\ldots,p'_n\}$, which are disjoint.
- Determine the area set $A = \{A_1,\ldots,A_n\}$. For each predicate $p'_i \in P'$, there is an area $A_i \in A$, where A corresponds to the horizontal fragmentation of R; in other words, A verifies the completeness constraints, which implies that predicates sub-expressions are not lost.
- Intersect every area $A_i \in A$ with partitions $T_1,\ldots,T_k$ to obtain a new set of subareas $\{S_{i1},\ldots,S_{1i_1}\}$. This new set corresponds to the vertical fragmentation of A_i; in other words, it verifies the completeness criteria.

Consequently, each hybrid fragmentation of a relation that satisfies the correctness constraints can be reorganized in areas and subareas; particularly, the fragments generated by Rakov's algorithm. Figure 2 shows an example of a reorganization.



(a) Before Reorganization



(b) After Reorganization

Figure 2: Fragments Reorganization.

To fragment database relations, we suggest applying Rakov's algorithm. Nevertheless, the fragmentation can be done manually by taking advantage of knowledge concerning the database relations collected from the user, DIS expert, or IT administrator.

Rakov's algorithm calculates the area accuracy as a percentage of accurate cells. Equation 6 calculates the Gini indices:

$$G(V) = 2k(1-k) \quad (6)$$

where k is the accuracy value, and the indices will be used to estimate the accuracy uniformity, and thereafter, to select the fragment representing the highest value of uniformity (i.e., the fragmentation function is parameterized to compute the area accuracy). Each fragment is reorganized as illustrated earlier. The area and subarea schema are also deduced from the related fragment using the following steps:

1. Name the areas sequentially and then their associated subareas.
2. Define the area predicate and the list of attributes describing subareas from the fragment.

3. Estimate the subarea accuracy by calculating the cell accuracy average (computed by Rakov's algorithm).

4. Estimate the key accuracy by calculating the average of accuracies of attributes conforming to the key (the accuracy of attributes is equal to their subarea accuracy owing to accuracy uniformity). Note that the key accuracy corresponds to the subarea accuracy if all key attributes are projected in a single subarea.

5. At the end, as Rakov's algorithm fragments a sample of database relation, the number of tuples in the sample is used to deduce the number of tuples in an area, i.e., $number\ of\ tuples \times \frac{relation\ size}{sample\ size}$.

The fragmentation is executed one time at DIS development or regularly. However, it is not related to the accuracy estimation of the query. The following section describes the query reformulation and its impact on accuracy estimation.

3.2 User query rewriting

Our approach proposes to reformulate the user query according to database relation areas. We will take into consideration the option of reformulation of user query in reference to subareas and we clarify the reason for discarding this alternative.

To describe a user query based on areas that constitute database relations, our approach uses the classic rewriting algorithm, i.e., the bucket algorithm. Hence, the LAV technique is applied to define the areas as views of a global schema by replacing the database relation with its description over the global schema, i.e., it unfolds views over the database relation. Note that this operation is unrelated to the user query because it is performed after fragmentation of database relations.

The query rewriting applies the bucket algorithm that creates buckets, compares predicates of the query and the area, and then determines possible rewritings and checks query containment. The number of rewritings increases with the number of areas in a polynomial manner. Note that the rigidity of the rewriting algorithm is not due to the number of relations, but to the query size (which is approximately small) and exists only if the query has more than one occurrence of the same relations [13].

Now, we discuss the option to rewrite a query over subareas and we explain why this option is rejected. Areas contain all tuples of database relations. Consequently, the rewriting algorithm of a query over areas joins all tuples of database relations (similar to query rewriting over database relations). Nevertheless, a subarea decomposes tuples as it projects a part of attributes of the relation. First, some versions of the bucket algorithm generate the total required attributes by joining multiple relations of a bucket. Consequently, query rewriting over subareas is not impossible. On the other hand, assembling a small number of attributes belonging to a large number of relations augments hazardously the risk of interpolating semantic inaccuracies (generation of tuples without meaning in real world). By way of illustration, the output can be the student name and the phone number of a different student who possess identical identifiers in distinct databases. This problem is intrinsic to DIS, but it remarkably grows if the number of joins increases (particularly when tuples are split). Moreover, the rewriting algorithm avoids splitting tuples specifically to reduce this risk.

Accordingly, our approach rewrites the query over areas and inevitably uses a subarea structure for computing data accuracy. This also explains why the proposed approach fragments relations in a horizontal and then vertical manner rather than by random management of hybrid fragmentations as Rakov's approach. The following section describes the estimation of data accuracy for each rewriting.

3.3 Data accuracy estimation

To compute area accuracies and aggregate them to calculate the accuracy value of the entire user query output, the proposed approach proceeds in three steps:

– identification of areas and subareas constituting the rewriting,

– estimation of the rewriting accuracy and its key accuracy, and

– calculation of the number of tuples in each area to perform the aggregation.

Identification of areas and subareas: An area contains at least one subarea and the rewriting is expressed as the join of area sets. The joining output is a unique area, which consists of the union of all subareas possessing part of the projected attributes and verifies the selection predicates of all input areas. The following example shows three areas, namely P_1, P_2, and P_3, decomposed to subareas (P_{11}, P_{12}), (P_{21},P_{22}), and (P_{31},P_{32},P_{33}) respectively (see Figure 3). The rewriting QR is defined as one area joining all subareas, which include part of the projected attributes. Note that, P_{12} is not included in QR because it does not contain any of the projected attributes in QR.

Because all tuples in the output of the rewriting fulfill all area predicates, a single area is built for the rewriting by joining the area predicates and rewriting predicates. In addition, because the bucket algorithm generates only relevant rewritings, the predicates cannot be contradictory. Note that we list only the predicates that are more constraining than others (e.g., "$age \geq 18$" is less constraining than "$age = 20$").

The rewriting subareas are defined as the aggregation of all subareas belonging to input areas, by considering just the common attributes between the subarea and rewriting. If the intersection between the rewriting and subarea is null, the subarea is discarded. The resulting rewriting

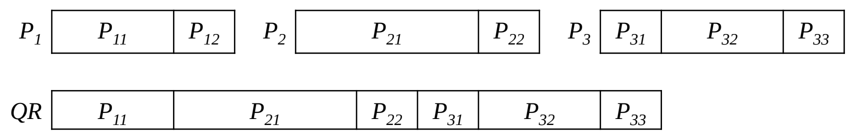


Figure 3: Rewriting Joins Multiple Areas.

subareas represent the vertical fragmentation of the rewriting area. The completeness constraint is satisfied because rewriting attributes appear in certain subareas of the input areas. However, for the disjunction constraint, the natural join emerges as a problem related to joining attributes existing in two different input subareas and appear only once in the rewriting. To solve this problem, a new subarea is created and its accuracy value will be computed as the average of both subarea accuracies (this will be discussed later).

Estimation of rewriting accuracy: If fragments are adequately determined, they logically conserve the accuracy of subareas because they are affected by the projection of predicates and attributes (owing to the accuracy uniformity). However, for the joining operation, each tuple in the subarea is composed of two input tuples; consequently, the produced subarea accuracy depends on both input areas. Particularly, the accuracy propagation can be different and depends on the accuracy factor. Recall that during the evaluation of semantic correctness, if the key of a particular tuple is inaccurate (does not reflect the real–world object), the entire tuple is considered as incorrect. Actually, semantic correctness evaluates the correspondence of an attribute (the key) to the real–world entity. On the other hand, the syntactic correctness evaluates the cell accuracy without regard to the key attributes. Thus, during area joining, the cell accuracy is computed differently from semantic correctness. Consequently, the subarea accuracy in the join output is computed as the product of the input subarea accuracy and key accuracy of the remaining areas. However, during syntactic correctness evaluation, the cell accuracy is equal to the cell accuracy of the input subarea.

Thus, our approach proceeds as follows:

– *Semantic correctness*: subarea accuracy is computed as the product of the input subarea accuracy and the key accuracy of the remaining areas.

– *Syntactic correctness*: subarea accuracy is calculated as the input subarea accuracy.

In the case of subareas that include join attributes (which were separated in the prior step), we calculate the accuracy for each input subarea separately and then take the average. The subareas with identical accuracy values are merged in a single subarea.

At the end, the accuracy of the key is calculated by multiplying the accuracy of keys in all areas. Then, the rewriting accuracy is calculated as the average of the cell accuracies (weighted average of subarea accuracy, where the weight is the number of attributes in the subarea).

Selectivity estimation: Because the query result is determined as the fusion of multiple rewritings, its accuracy is estimated as the weighted aggregation of the rewriting accuracy, where the weight is equal to the rewriting size (number of tuples in the rewriting). Thus, the estimation of the number of tuples in each rewriting is necessary.

To this end, we suggest estimating the rewriting selectivity to deduce its number of tuples. In general, a join corresponds to the comparison between two keys (primary and foreign), and then histograms are used for the calculation. Note that query optimization algorithms or statistical information about the prior execution of the same/similar query can be applied to estimate the selectivity. Particularly, experts also can estimate the selectivity. Remember that our approach does not depend on the estimation algorithm but certainly, the resulting accuracy value depends on it. The selectivity is defined as follows:

Given a query rewriting Q_R over areas $\{R_1, \ldots, R_k\}$, the selectivity of Q_R, expressed as $S(Q_R)$, represents the number of tuples in the Cartesian product of all areas that satisfy the rewriting predicates. If Q_R does not have a selection (or join) predicate, the whole set of tuples is considered and in this manner, $S(Q_R) = 1$. Concerning the rewriting query, the number of tuples is calculated as:

$$Number\ of\ tuples\ in\ the\ input\ areas\ \times\ Rewriting\ selectivity$$

3.4 Quality graph

After generation of the query rewritings, we build a quality graph to calculate the user query as the union of all rewritings [10, 18]. We can have multiple rewritings or a unique one.

The creation of the quality graph shown in Figure 4 follows these definitions:

– *Source* nodes represent database relations.

– *Target* node represents user query.

– Activity nodes represent areas of database relations, denoted as Area. Edges connect the area to its source.

– Activity nodes represent the rewritings, denoted by *Rewriting*. Edges connect the rewriting to the area indicated by the rewriting.

– One activity node represents the union of rewritings (possibly single rewriting), denoted by the *Union* node. This node is the successor of all rewriting nodes and the predecessor of the *Target* node.

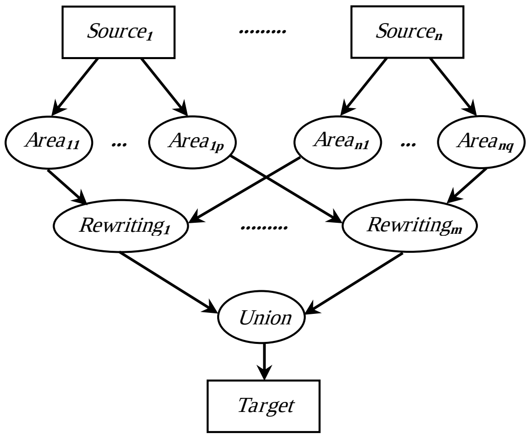


Figure 4: Quality Graph.

3.5 Accuracy estimation algorithm

We propose an algorithm for data accuracy estimation. The pseudo–code is shown in Figure 5. It implements the approach previously described (section 3.3) consisting of three phases:

1. For each *Rewriting*, the algorithm generates an area, defines its subareas, and computes the characteristic values (cardinality, predicates, subarea accuracy, and key accuracy).

2. The algorithm assembles all rewritings to determine the *Union* node.

3. The algorithm performs the accuracy value aggregation for all data edges.

In the first phase, the algorithm executes two loops over area nodes that are inputted to each rewriting node. During the first loop, the key accuracy feature is calculated by multiplying the accuracy of all areas; the predicate feature is determined by merging the predicates of input areas and the rewriting predicate, and eliminating worthless constraints; finally, the cardinality feature is computed as the product of the rewriting selectivity and the input area cardinalities.

During the second loop, to estimate the subarea feature, we just insert subareas of all input areas. This should be performed in another loop as the input of subarea accuracy is the key accuracy computed in the first loop. Then the algorithm adds subareas to the rewriting node and joins subarea attributes with the rewriting attributes. The natural join attributes are stored in new subareas. We calculate their accuracy using equation 7:

$$Accuracy = \frac{\sum_1^n subarea_i}{n} \tag{7}$$

where n is the number of subareas. At the end, the subareas with identical accuracy values are merged. U_{areas} is a list that contains all generated areas for each rewriting; consequently, the second phase requires setting U_{areas} as the value of the union node.

Finally, for each source or activity node, the accuracy aggregation is calculated as the weighted sum of the subarea accuracy and the weight is calculated by multiplying the number of subarea attributes by the area cardinality. The resulting output is associated with all edges outgoing the node.

4 Case study

This example is used to demonstrate the proposed approach for data accuracy estimation. The Boolean metric is used to illustrate the evaluation of semantic correctness; nevertheless, alternative accuracy metrics and factors can be used.

Assume that the DIS global schema contains two relations that hold data concerning students and their marks respectively:

– $Student(ID, name, level, exam, phone, address, city)$

– $Mark(ID, mark, year)$

Attributes of the relation student are ID (the student identifier), $name$, $level$ (first level defined by initial interview and its value can be "low", "medium", or "high"), $exam$ (initial exam result; its value $\in$ $[0, 1]$), $phone$, $address$, and $city$.

Attributes symbolizing the relation mark are ID (the student identifier), $mark \in [0 - 20]$, where 20 is the highest mark), and $year$. The relation keys are ID and $ID, year$ respectively.

Suppose that two databases provide data concerning students and their marks as presented in Table 1 and Table 2:

– $S(ID, name, level, exam, phone, address)$ // students residing at Al–Ain (UAE).

– $M(ID, mark, year)$ // the student marks.

The colored cells represent the inaccuracy. The aggregated accuracy values for S and M relations (calculated as the average of cell accuracies) are 0.6 $(40/66)$ and 0.77 $(37/48)$ respectively. The key expansions are:

$$\bar{S}(K_{ID}, ID, name, level, exam, phone, address)$$

and

$$\bar{M}(K_{ID}, K_{year}, ID, mark, year)$$

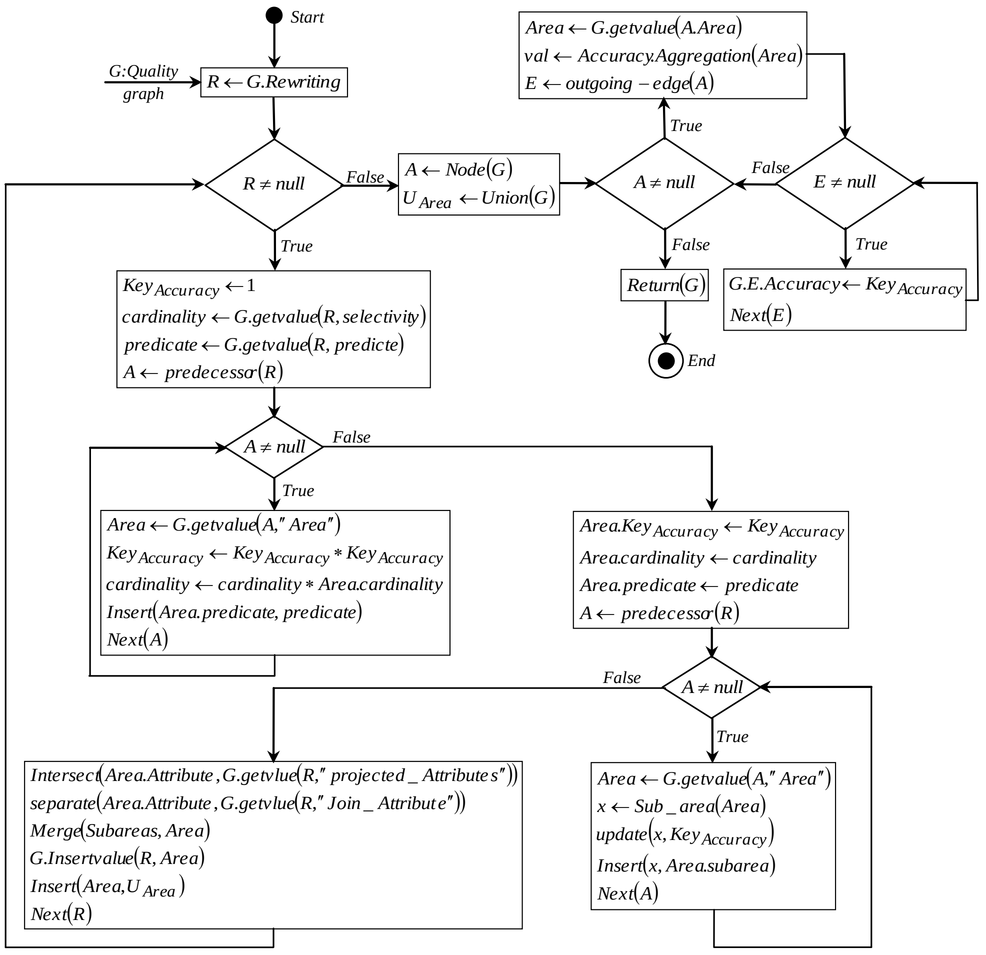

Start
G:Quality graph
R ← G.Rewriting
R ≠ null
False
True
A ← Node(G)
U_Area ← Union(G)
Key_Accuracy ← 1
cardinality ← G.getvalue(R, selectivity)
predicate ← G.getvalue(R, predicte)
A ← predecessor(R)
A ≠ null
Area ← G.getvalue(A,"Area")
Key_Accuracy ← Key_Accuracy * Key_Accuracy
cardinality ← cardinality * Area.cardinality
Insert(Area.predicate, predicate)
Next(A)
Area.Key_Accuracy ← Key_Accuracy
Area.cardinality ← cardinality
Area.predicate ← predicate
A ← predecessor(R)
Area ← G.getvalue(A,"Area")
x ← Sub_area(Area)
update(x, Key_Accuracy)
Insert(x, Area.subarea)
Next(A)
Intersect(Area.Attribute, G.getvlue(R,"projected_Attributes"))
separate(Area.Attribute, G.getvlue(R,"Join_Attribute"))
Merge(Subareas, Area)
G.Insertvalue(R, Area)
Insert(Area, U_Area)
Next(R)
Area ← G.getvalue(A.Area)
val ← Accuracy.Aggregation(Area)
E ← outgoing − edge(A)
E ≠ null
Return(G)
End
G.E.Accuracy ← Key_Accuracy
Next(E)

Figure 5: Accuracy Propagation Algorithm.

where $\{K_{ID}\}$ and $\{K_{year}, K_{ID}\}$ define the expanded key. Now S and M are fragmented to determine their areas and subareas; then, we calculate the number of tuples and their accuracies. An acceptable fragmentation of S is:

- Area S_1; $[ID < 300]$; 6 tuples; $keyaccuracy = 0.50$
  - Subarea S_{11}; $\{ID, name, level, exam\}$; $accuracy = 0.50$
  - Subarea S_{12}; $\{address, phone\}$; $accuracy = 0.25$
- Area S_2; $[ID \geq 300]$; 5 tuples; $keyaccuracy = 1.00$
  - Subarea S_{21}; $\{ID\}$; $accuracy = 1.00$
  - Subarea S_{22}; $\{name, level, exam, phone, address\}$; $accuracy = 0.80$

Similarly, a possible fragmentation of the M relation is:

- Area M_1; $[year < 2014]$; 1 tuple; $keyaccuracy = 0$
  - Subarea M_{11}; $\{ID, mark, year\}$; $accuracy = 0.00\ (0/3)$
- Area M_2; $[year \geq 2014 \wedge id < 300]$; 6 tuples; $keyaccuracy = 0.50$
  - Subarea M_{21}; $\{ID, mark, year\}$; $accuracy = 0.50\ (9/18)$
- Area M_3; $[year \geq 2014 \wedge ID \geq 300]$; 8 tuples; $keyaccuracy = 0.93$
  - Subarea M_{31}; $\{ID, mark, year\}$; $accuracy = 0.93\ (25/27)$

After horizontal and vertical fragmentation of the relation, each cell belongs to a specific subarea included in a unique area; consequently, the fragments are represented by different colors describing different subareas. Table 3 presents the fragments of the student relation. The expanded key is not colored because it belongs to all subareas and it can be left out in the graphical illustration.
The respective schemas of S and M are:

- $S(ID, name, level, exam, phone, address, city) \leftarrow Student(ID, name, level, exam, phone, address) \wedge city = \text{“}Al - Ain''$

Table 1: S Relation.

ID	Name	Level	Exam	Phone	Address
120	Rani	High	1	6001104	12
123	Nada	Medium	0.465	99628734	Chiab al Alashekhar
141	Areej	Low	0.987		Al–Ain
154	Hanan	Low	0.1234	9023365	Palmier 13
155	Zineb	Medium	0.61	3364244	502 logts 13 n4
157	Deena	Low	0.2	7091232	Annaba 69
300	Assala	High	0.97	4112533	
301	Alae	High	0.92	5437898	
302	Raid	High	0.78		Abu Dhabi
303	Med	Low	0.2	3248673	Algeria 1280/12
304	Sarah	Medium	0.67	231987253	

Table 2: M Relation.

ID	Mark	Year
120	17	2015
141	10	2015
141	18	1014
154	9	2014
155	13	2014
155	4	2015
157	5	2015
300	10	2014
300	19	2015
301	17	2014
301	12	2014
302	18	2015
302	6	2014
303	9	2015
304	11	2014
304	13	2015

– $M(ID, mark, year) \leftarrow Mark(ID, mark, year)$

The areas of S and M are expressed in datalog–like notation:

– $S_1(ID, name, level, exam, phone, address) \leftarrow S(ID, name, level, exam, phone, address) \wedge ID < 300$

– $S_2(ID, name, level, exam, phone, address) \leftarrow S(ID, name, level, exam, phone, address) \wedge ID \geq 300$

– $M_1(ID, mark, year) \leftarrow M(ID, mark, year) \wedge y < 2014$

– $M_2(ID, mark, year) \leftarrow M(ID, mark, year) \wedge y \geq 2014 \wedge ID < 300$

– $M_3(ID, mark, year) \leftarrow M(ID, mark, year) \wedge y \geq 2014 \wedge ID \geq 300$

The substitution of S and M by their expressions results in the area definition in terms of the global schema:

– $S_1(ID, name, level, exam, phone, address) \leftarrow Student(ID, name, level, exam, phone, address) \wedge city = \text{“}Al-Ain'' \wedge ID < 300$

– $S_2(ID, name, level, exam, phone, address) \leftarrow Student(ID, name, level, exam, phone, address) \wedge city = \text{“}Al-Ain'' \wedge ID \geq 300$

– $M_1(ID, mark, year) \leftarrow Mark(ID, y, m) \wedge y < 2014$

– $M_2(ID, mark, year) \leftarrow Mark(ID, y, m) \wedge y \geq 2014 \wedge ID < 300$

– $M_3(ID, mark, year) \leftarrow Mark(ID, y, m) \wedge y \geq 2014 \wedge ID \geq 300$

Suppose the user query Q in datalog–like notation:

– $Q(ID, name, year, mark) \leftarrow S(ID, name, level, exam, phone, address, city) \wedge M(ID, mark, year) \wedge y = 2015$

The output of query Q (extracted from the S and M relations) is given in Table 4.
The query rewriting using S_1, S_2, M_1, M_2, and M_3 creates the following buckets: $Buck(S) = \{S_1, S_2\}$ and $Buck(M) = \{M_2, M_3\}$. The area M_1 is omitted in $Buck(M)$ because it does not satisfy the Q predicate (“$year = 2015$” in Q and “$year < 2014$” in M_1).
Then the query rewritings are produced, considering one area for each bucket:

– $QR_1(ID, year, name, mark) \leftarrow S_1(ID, name, level, exam, phone, address) \wedge M_2(ID, mark, year) \wedge y = 2015$

Table 3: Student Relation Fragmentation.

KID	ID	Name	Level	Exam	Phone	Address
120	120	Rani	High	1	6001104	Zakhir 12
123	123	Nada	Medium	0.465	99628734	Chiab al Alashekhar
141	141	Areej	Low	0.987		Al–Ain
154	154	Hanan	Low	0.1234	9023365	Palmier 13
155	155	Zineb	Medium	0.61	3364244	502 logts 13 n4
157	157	Deena	Low	0.2	7091232	Annaba 69
300	300	Assala	High	0.97	4112533	
301	301	Alae	High	0.92	5437898	
302	302	Raid	High	0.78		Abu Dhabi
303	303	Med	Low	0.2	3248673	Algeria 1280/12
304	304	Sarah	Medium	0.67	231987253	

Table 4: Query Result.

ID	Name	Year	Mark
120	Rani	2015	17
141	Areej	2015	10
155	Zineb	2015	4
157	Deena	2015	5
300	Assala	2015	19
302	Raid	2015	18
303	Med	2015	9
304	Sarah	2015	13

- $QR_2(ID, year, name, mark) \leftarrow S_2(ID, name, level, exam, phone, address) \wedge M_3(ID, mark, year) \wedge y = 2015$

Because the areas, S_1 and M_3 are contradictory ("$ID \geq 300$" and "$ID < 300$"), they are not joined; the same applies for areas S_2 and M_2. Consequently, $Q \supseteq QR_1 \bigcup QR_2$.
Let us focus on the rewriting of QR_1:
QR_1 and QR_2 areas and subareas are defined as creations of a unique area for each rewriting of QR_1 and QR_2 respectively, because the rewriting of QR_1 combines areas S_1(its subareas are S_{11}, S_{12}) and M_2(its subarea is M_{21}). The intersection of subarea S_{11} and M_{21} attributes with the rewriting attributes results in two subareas QR_{11} and QR_{12}; the join attribute is isolated in subarea QR_{13}. In this case there is no projection of the subarea S_{12} attribute. QR_2 subareas are defined similarly. The final result is:

- Area $QR_1\,\{ID < 300 \wedge year = 2015\}$; ? tuples; $keyaccuracy$ =?; inputs: S_1, M_2
  - Subarea $QR_{11}\,\{name\}$; $accuracy$ =?; input: S_{11}
  - Subarea $QR_{12}\,\{year, mark\}$; $accuracy$ =?; input: M_{21}
  - Subarea $QR_{13}\,\{ID\}$; $accuracy$ =?; inputs: S_{11} and M_{21}
- Area $QR_2\,\{ID \geq 300 \wedge year = 2015\}$; ? tuples; $keyaccuracy$ =?; inputs: S_2, M_3
  - Subarea $QR_{21}\,\{name\}$; $accuracy$ =?; input: S_{22}
  - Subarea $QR_{22}\,\{year, mark\}$; $accuracy$ =?; input: M_{31}
  - Subarea $QR_{23}\,\{ID\}$; $accuracy$ =?; inputs: S_{21}, M_{31}

After the estimation of semantic accuracy of subareas, we obtain

- Area $QR_1\,\{ID < 300 \wedge year = 2015\}$; ? tuples; $keyaccuracy = 0.25\ (0.50 \times 0.50)$; inputs: S, M_2
  - Subarea $QR_{11}\,\{name\}$; $accuracy = 0.25 = (averageof(0.50 \times 0.50))$; input: S_{11}
  - Subarea $QR_{12}\,\{year, mark\}$; $accuracy = 0.25 = 0.50 \times 0.50$; input: M_{21}
  - Subarea $QR_{13}\,\{ID\}$; $accuracy = 0.25(average(0.50 \times 0.50, 0.50 \times 0.50)$; input: S_{11}, M_{21}
- Area $QR_2\,\{ID \geq 300 \wedge year = 2015\}$; ? tuples; $keyaccuracy = 0.93(1.00 \times 0.93)$; inputs: S_2, M_3
  - Subarea $QR_{21}\,\{name\}$; $accuracy = 0.74(0.80 \times 0.93)$; input: S_{22}
  - Subarea $QR_{22}\,\{year, mark\}$; $accuracy = 0.93(0.93 \times 1.00)$; input: M_{31}
  - Subarea $QR_{23}\,\{ID\}$; $accuracy = 0.84(average0.8 \times 0.93, 0.93 \times 1.00)$; inputs: S_{21}, M_{31}

Because subareas QR_{11}, QR_{12}, QR_{13} possess the same accuracy, they are merged to subarea QR_{11}.

- Area $QR_1\,\{ID < 300 \wedge year = 2015\}$; ? tuples; $keyaccuracy = 0.25$; inputs: S_1, M_2
  - Subarea $QR_{11}\,\{ID, name, year, mark\}$; $accuracy = 0.25$
- Area $QR_2\,\{ID{\geq}300 \wedge year = 2015\}$; ? tuples; $keyaccuracy = 0.93$; inputs: S_2, M_3
  - Subarea $QR_{21}\,\{name\}$; $accuracy = 0.74$; input: S_{22}
  - Subarea $QR_{22}\,\{year, mark\}$; $accuracy = 0.93$; input: M_{31}
  - Subarea $QR_{23}\,\{ID\}$; $accuracy = 0.84$; inputs: S_{21} and M_{31}

Remember that the rewriting accuracy is the average of the cell accuracies (accuracy aggregation). In this manner, the QR_1 accuracy is 0.25 and the QR_2 (which represents the accuracy value of its unique subarea) accuracy is $0.86(0.74 \times 1 + 0.93 \times 2 + 1.00 \times 3)$.
Thus, the QR_1 and QR_2 selectivity is calculated as $\frac{1}{9}(\frac{4}{6\times6})$ and $\frac{1}{10}(\frac{4}{5\times8})$ respectively. Subsequently, the partition metadata is

- Area $QR_1\,\{ID < 300 \wedge year = 2015\}$; $\frac{1}{9}(6{\times}6) = 4$ tuples; $keyaccuracy = 0.25$; inputs: S_1, M_2
  - Subarea $QR_{11}\,\{ID, name, year, mark\}$; $accuracy = 0.25$
- Area $QR_2\,\{ID{\geq}300 \wedge year = 2015\}$; $4(= 5 \times 8 \times 1/10)$ tuples; $keyaccuracy = 0.93$; inputs: S_2, M_3
  - Subarea $QR_{21}\,\{name\}$; $accuracy = 0.74$
  - Subarea $QR_{22}\,\{year, mark\}$; $accuracy = 0.93$
  - Subarea $QR_{23}\,\{ID\}$; $accuracy = 0.84$

The global user query Q accuracy value is estimated as the sum of the weighted and rewriting accuracies ($weight = tuplenumbers$); thus, we obtain $(4{\times}0.25{+}4{\times}0.93)/(4{+}4) = 0.59$.
Figure 6 shows the quality graph for user query Q defined by merging its rewritings QR_1 and QR_2.

In the following section, we will discuss some data accuracy improvements by presenting a possible approach, which discards the areas or subareas causing exaggeration in accuracy prediction.

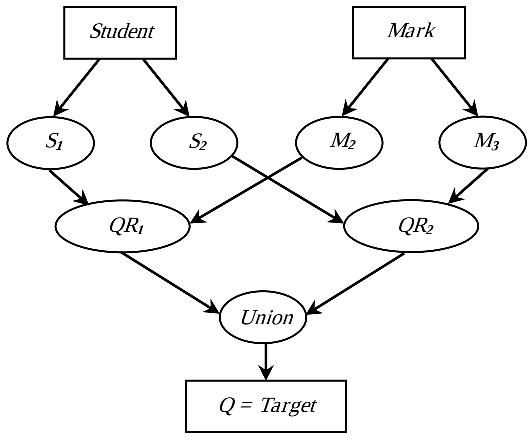


Figure 6: Quality Graph.

5 Accuracy improvement

To enforce the data accuracy, some fundamental amendment actions can be taken if the user expectations on data accuracy cannot be achieved. Because the proposed estimation approach arranges the query outputs in subareas possessing uniform accuracy, there is no guarantee that the total cells in the query output satisfy the target expectations. However, the delivered data, in general, satisfy the user target.

We propose a simple improvement action consisting of discarding pieces of the query output with small accuracy values. We will present different ways and different times of executing such improvement actions. Particularly, we distinguish three accuracy levels:

- *Cell level*: Accuracy of cell set, in general, must be less than a specific threshold.
- *Tuple level*: Accuracy of tuple set, in general, must be less than a specific threshold.
- *Output level*: Accuracy of the whole output, must be less than a specific threshold.

The *cell level* is the most constraining. It reflects the fact that the user accepts only tuples containing accurate values. Moreover, the accuracy restrictions may be related to specified attributes; as an example, the user requires accurate phone numbers and disregards the accuracy of other attributes. For this type of restriction, we suggest filtering subareas with small accuracy values because the query output is partitioned to subareas with uniform accuracy. If the restriction concerns only specific attributes, filtering concerns only the subareas involving those attributes. Such action can be performed in the initial evaluation phase, exactly during insertion of areas in the buckets. We call this improvement action selective rewriting.

The *tuple level* accepts the existence of a group of attributes having low accuracy with the condition that the aggregate accuracy of the tuple is sufficiently high. Some users may tolerate acquiring tuples containing inaccuracies in certain attributes as long as the remaining cells are correct (e.g., if different attributes hold possible manner to contact clients such as phone, address, and email). Therefore, mistakes in some attributes are tolerable if the remaining attributes have high accuracy values. In this case, we suggest area filtering rather than subarea filtering; in other words, the accuracy value is the aggregation of subarea accuracy, which is then compared to user expectations. Note that area filtering should be performed after rewriting computation (contrary to subarea filtering), as the aggregation is communicated to several subareas belonging to different database relations.

Finally, the *output level* does not consider the tuple or attribute accuracy; however, it signifies that the final output must reach a specific accuracy level. Remember that query outputs containing a mixture of tuples having high accuracy and tuples having very low accuracy are tolerable. This reflects people requesting all possible output data, without ignoring accuracy (e.g., the user sends publicity message to clients, accepting up to 6% undelivered message because of mistakes in contact attribute). To satisfy this constraint type, we must generate all possible data without considering their accuracy. To this end, we suggest sorting areas by their accuracies, aggregate the accuracy values gradually, and then stop if the correctness constraints are unsatisfied. Additionally, the data delivery can be incremental, allowing the users to stop when they obtain the necessary data or data containing many mistakes. This approach is executed only after aggregating the accuracy of all rewritings.

Therefore, three improvement actions are proposed:

- *Selective rewriting* is the generation of query rewritings that achieve a specific quality level, particularly "all subarea accuracy should be greater than the threshold." In this case, the rewriting algorithm can be modified to exclude from buckets the areas related to subareas having low accuracy. As the subarea accuracy was precomputed during the fragmentation process, this strategy can be implemented easily.

- *Rewriting filtering* can be implemented directly during the aggregation of rewriting accuracy.

- *Incremental data delivery*. After the aggregation of rewriting accuracy, we suggest sorting them according to descending order of accuracies. The algorithm is shown in Figure 7. The inputs of the algorithm are the expected values of accuracy and a list of areas in the rewritings; the output is a sorted area list. The list represents the data that satisfy user expectations.

Note that if the constraints are more restrictive, the output size will be small. It is worthy to mention that we should have a balance between accuracy and completeness expectations to deliver useful and valuable tuples to the user and avoid filtering too much data.

Figure 8 shows an example that illustrates the proposed improvement actions. Assume that QR_1, QR_2, QR_3, and QR_4 represent four rewritings. The number inside each subarea indicates the accuracy value of that subarea and the aggregated accuracy value for each area is written in front of QR_i.

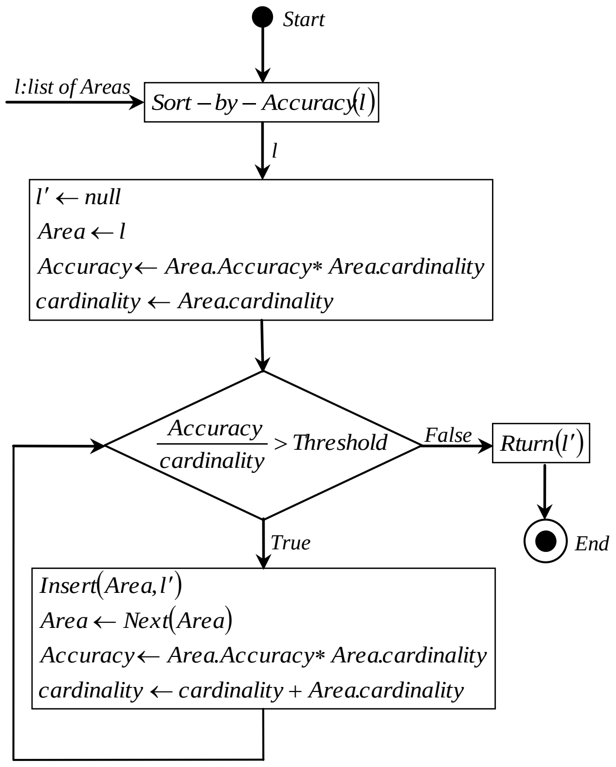

Start
l:list of Areas
Sort – by – Accuracy(l)
l
l′ ← null
Area ← l
Accuracy ← Area.Accuracy * Area.cardinality
cardinality ← Area.cardinality
Accuracy / cardinality > Threshold
False
Rturn(l′)
End
True
Insert(Area, l′)
Area ← Next(Area)
Accuracy ← Area.Accuracy * Area.cardinality
cardinality ← cardinality + Area.cardinality

Figure 7: Incremental Data Delivery.

Assume also that C_1, C_2, and C_3 describe three constraints as follows:

- C_1: All subarea accuracy values must be more than or equal to 0.8.

- C_2: All area accuracy values must be more than 0.8.

- C_3: The query accuracy value must be more than 0.7.

Only QR_3 satisfies constraint C_1. Thus, the algorithm will generate only QR_3. Furthermore, QR_4 will be discarded because one of its subareas S_{41} does not satisfy the constraint. However, QR_3 and QR_4 satisfy constraint C_2. Despite the fact that subarea S_{41} possesses a low accuracy value than 0.8, the whole area accuracy is accepted.
To check constraint C_3, we sort QR_i values by accuracy. Thus, we obtain this order: $QR_3 \rightarrow QR_4 \rightarrow QR_2 \rightarrow QR_1$. We compute the aggregate accuracy gradually:

- $\{QR_3\}$ accuracy is 0.9.

- $\{QR_3, QR_4\}$ accuracy is $(0.9 \times 20 + 0.8 \times 10)/(20 + 10) = 0.86$.

- $\{QR_3, QR_4, QR_2\}$ accuracy is $(0.9 \times 20 + 0.8 \times 10 + 0.6 \times 10)/(20 + 10 + 10) = 0.8$.

- $\{QR_3, QR_4, QR_2, QR_1\}$ accuracy is $(0.9 \times 20 + 0.8 \times 10 + 0.6 \times 10 + 0.5 \times 20)/(20 + 10 + 10 + 20) = 0.7$. Because constraint C_3 is not satisfied, QR_1 is discarded.

QR_1	S_{11}	S_{12}	S_{13}	S_{14}	20 tuples
0.5	0.35	0.49	0.51	0.59	

QR_2	S_{21}	S_{22}	10 tuples
0.6	0.47	0.64	

QR_3	S_{31}	S_{32}	S_{33}	20 tuples
0.9	0.8	0.97	0.93	

QR_4	S_{41}	S_{42}	S_{43}	S_{44}	S_{45}	S_{46}	10 tuples
0.8	0.4	0.92	0.82	0.85	0.93	0.85	

Figure 8: Rewritings Filtering.

6 Conclusion

This study investigates the data accuracy estimation and proposes some improvement actions. We suggested the fragmentation of database relations according to data accuracy by applying Rakov's algorithm and the projection of such fragments to query outputs to report the inaccuracy distribution in a better way. We rewrote the user query according to the fragments, and then we aggregated the accuracy values for the rewritings. We defined the query output by uniting the output tuples of rewritings. A basic algorithm to estimate the data accuracy was presented. In contrast to the existing approaches, the areas with low accuracy are identified explicitly by our proposed algorithm, so that data unsatisfying user expectations are discarded.

Furthermore, we suggested three primary improvement actions in order to filter data having lower accuracy to meet different types of user expectation accuracy. This approach is applicable in multiple stages of DIS development (design, production, or maintenance) to inform users about data accuracy, compare databases, check expectation satisfaction, or analyze enforcement actions for improving data accuracy.

Our objective in the near future is the development of a prototype implementing the proposed accuracy evaluation algorithm and the illustration of its usage in real–world applications. For this purpose, the prototype should display and edit the different components (such as databases, quality graph, characteristics, accuracy values, etc.) of the framework in addition to the execution of the quality evaluation algorithms. Furthermore, the prototype should evaluate the data accuracy in different applications for validation purposes by describing several tests in order to assess the approach performance and limitations. Finally, we suggest to improve some features of the quality evaluation tool and perform additional performance tests. In addition, we aim to analyze further quality factors and their inter–relationships.

References

[1] Abdalla, H., and Artoli, A. M. Towards an efficient data fragmentation, allocation, and clustering approach in a distributed environment. *Information*, 10(3), 112, 2019. DOI: 10.3390/info10030112

[2] Agrawal, S., Narasayya, V., and Yang, B. Integrating vertical and horizontal partitioning into automated physical database design. *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 359–370, 2004. DOI: 10.1145/1007568.1007609

[3] Alsaif, S. A., and Hidri, A. Impact of Data Balancing During Training for Best Predictions. *Informatica*, 45(2), 2021. DOI: 10.31449/inf.v45i2.3479

[4] Azeroual, O., Ershadi, M. J., Azizi, A., Banihashemi, M., and Abadi, R. E. (2021). Data Quality Strategy Selection in CRIS: Using a Hybrid Method of SWOT and BWM. *Informatica*, 45(1), 2021. DOI: 10.31449/inf.v45i1.2995

[5] Bai, Q., Hong, J., and McTear, M. F. Some modifications of bucket-based algorithms for query rewriting using views. *In International Conference on Advances in Information Systems*, Springer, Berlin, Heidelberg, 57–67, 2004. URL: https://rb.gy/leqpjc

[6] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. Classification and Regression Trees, *: Wadsworth*, 368, 1984. DOI: 10.1201/9781315139470-8

[7] Calvanese, D., Lembo, D., and Lenzerini, M. Survey on methods for query rewriting and query answering using views. *Integrazione, Warehousing e Mining di sorgenti eterogenee*, 25, 2001. URL: https://rb.gy/05cken

[8] Ezéchiel, K. K., Kant, S., and Agarwal, R. A systematic review on distributed databases systems and their techniques. *Journal of Theoretical and Applied Information Technology*, 96(1), 236–266, 2019. URL: https://rb.gy/gs00qk

[9] Fuaad, H. A., Ibrahim, A. A., Majed, A., and Asem, A. A Survey on Distributed Database Fragmentation Allocation and Replication Algorithms. *Current Journal of Applied Science and Technology*, 27(2), 1–12, 2018. DOI: 10.9734/CJAST/2018/37079

[10] Gao, J., Li, X., Xu, Y. E., Sisman, B., Dong, X. L., and Yang, J. Efficient knowledge graph accuracy evaluation. *arXiv preprint arXiv:1907.09657*, 2019. DOI: 10.48550/arXiv.1907.09657

[11] Halevy, A. Y. Answering queries using views: A survey. *The VLDB Journal*, 10(4), 270–294, 2002. DOI: 10.1007/s007780100054

[12] Hill, G. A Framework for valuing the quality of Customer Information *Doctoral dissertation, Melbourne University*, 1–194, 2009. URL: http://hdl.handle.net/11343/35567

[13] Levy, A., Rajaraman, A., and Ordille, J. Querying heterogeneous information sources using source descriptions. *In Proceedings of 22th International Conference on Very Large Data Bases*, 1, 1–26, 1996. DOI: 10.1049/tpe.1981.0030

[14] Ma, H., Noack, R., Schewe, K. D., and Thalheim, B. Using meta–structures in database design. *Informatica*, 34(3), 387—403, 2010. https://rb.gy/cbonrz

[15] Motro, A., and Rakov, I. Estimating the quality of databases. *In International Conference on Flexible Query Answering Systems*, Springer, Berlin, Heidelberg, 298–307, 1998. URL: https://rb.gy/iriobk

[16] Naumann, F., Leser, U., and Freytag, J. C. Quality–driven integration of heterogeneous information systems. *In Proceedings of the International Conference on Very Large Data Bases*, 447–458, 2005. URL: https://rb.gy/bgzhyn

[17] Özsu, M. T., and Valduriez, P. Principles of Distributed Database Systems, *Management, Springer, New York*, 12, 657–722, 2011. DOI: 10.1007/978-1-4419-8834-8

[18] Peng, P., Zou, L., Chen, L., and Zhao, D. Adaptive distributed RDF graph fragmentation and allocation based on query workload. *IEEE Transactions on Knowledge and Data Engineering*, 31(4), 670–685, 2018. DOI: 10.1109/TKDE.2018.2841389

[19] Rakov, I. Data quality and its use for reconciling inconsistencies in multidatabase environment.*PhD dissertation, George Mason University*, 1998.

[20] Raman, V., Swart, G., Qiao, L., Reiss, F., Dialani, V., Kossmann, D., Narang, I. and Sidle, R. Constant–time query processing. *In 2008 IEEE 24th International Conference on Data Engineering*, 60–69, 2008. DOI: 10.1109/ICDE.2008.4497414

[21] Riezler, S., and Liu, Y. Query rewriting using monolingual statistical machine translation. *Computational Linguistics*, 36(3), 569–582, 2010. DOI: 10.1162/coli_a_00010

[22] Silberschatz, A., Korth, H. F., and Sudarshan, S. Database system concepts. *McGraw-Hill, New York*, 5, 2002. URL: https://rb.gy/ttrvjt

[23] Stvilia, B. A workbench for information quality evaluation, *In Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital libraries*, 469–469, 2008. DOI: 10.1145/1378889.1379014

[24] Tarun, S., Batth, R. S., and Kaur, S. A Review on Fragmentation, Allocation and Replication in Distributed Database Systems. *In International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 538–544, 2019. DOI: 10.1109/ICCIKE47802.2019.9004233

[25] Várkonyi, G. G., and Gradišek, A. Data protection impact assessment case study for a research project using artificial intelligence on patient data. *Informatica*, 44(4), 2020. DOI: 10.31449/inf.v44i4.3253

[26] Wang, R. Y., and Strong, D. M. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4), 5–33, 1996. DOI: 10.1080/07421222.1996.11518099

[27] Zozus, M. N., Pieper, C., Johnson, C. M., Johnson, T. R., Franklin, A., Smith, J., and Zhang, J. Factors affecting accuracy of data abstracted from medical records. *PloS one*, 10(10), e0138649, 2015. DOI: 10.1371/journal.pone.0138649