Contributions to Quality-Aware Online Query Processing

Laure Berti-Équille
IRISA, Campus Universitaire de Beaulieu, Rennes, France

Abstract

For non-collaborative data sources, quality-aware query processing is difficult to achieve because the sources generally do not export data quality indicators. This paper presents a prospective work on the declaration of metadata describing data quality and on the adaptation of query processing for taking into account constraints on the quality of data and finding dynamically the best trade-off between the cost of the query and the quality of the result.

1 Introduction

In both centralized or distributed query applications (e.g., in decision support area or Bussiness Intelligence), a set of interesting data sources may be potentially candidate to answer a query. But these sources are usually non-collaborative and do not export information describing the local cost of their query processing, neither indicators of their quality of service (e.g., resource accessibility, reliability, security, etc.), nor information describing the quality of their content (e.g., data accuracy, availability, freshness, completeness, etc.). Relational query optimization has traditionally relied upon table cardinalities when estimating the cost of query plans they consider. While this approach has been and continues to be successful, the need to consider the various dimensions of data quality (such as accuracy, freshness, completeness, etc.) for query execution requires a particular approach. The dual problem is to fix the query cost and search for the "best quality" result, or to fix the result quality and optimize the query cost.

Data quality awareness when querying single or several distributed data sources in a dynamic and distributed environment raises several interesting questions such as:

- Selecting dynamically the adequate data source: different data sources may answer a global query with different response times, query costs and various levels of data quality. How to define strategies for selecting adaptively the most appropriate sources for answering a query with suitable (or, at least, acceptable) data quality?
- Defining semantically and qualitatively correct distributed query plans: the result of a global query is classically built depending on the particular order for the execution of subquery plans. For ensuring data quality awareness, this technique must combine in a coherent way both information and meta-information from the various data sources (*i.e.*, data quality metadata if available). Data quality levels are often unknown, heterogeneous from one source to another, more or less aggregated or locally non uniform (*i.e.*,

---

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

a source may provide excellent data reliability for one specific area, data subset or data type but not for the others). In this context, one of the problems is to control and merge the data quality indicators in a consistent way.

- Making trade-offs between the query cost and the measurable result quality: because one may accept a query result of lower quality (if it is cheaper or has a shorter response time than if the query cost is higher), it's necessary to adapt the query cost to users' quality requirements. The objective is to measure and optimally reduce the cost and bargain query situations where the system searches for solutions that "squeeze out" more gains (in terms of data quality of the query result) than the query without data quality constraints.

- Developing query cost models to evaluate whether the expected benefits from a "quality-augmented" query compensate for the cost of computing or predicting quality indicators and collecting feedbacks from the source and the environment during execution time. The difficulty is to adapt existing query processing techniques to environments where resource availability, allocation, query cost and data quality may be not decidable at compile time.

Several "static" approaches have been proposed to select the data sources before the query processing; they mainly use metadata describing the source content, structure, and quality (**?**, **?**). The work presented in (**?**) studied the alternative distributed query plans for mediation systems. Naumann proposed a distributed query planning algorithm that enumerates query plans in such way that it usually finds the best N plans after computing only a fraction of the total number of plans. Upper quality bounds for partial query plans are constructed and thereby non-promising subplans are early pruned in the search tree. This technique relies on source-specific quality criteria and also query-specific quality criteria for the selection phases in the planning algorithm. In (**?**), Naumann extends this work and proposes mechanisms to follow the execution of the query and, if necessary, to cancel it or change the query plan execution. In (**?**), the author proposes to take into account data quality estimates when evaluating the users query and deciding the best manner of carrying out the query (which sources to reach, which server to use, etc).

To the best of our knowledge, the issues of data quality-awareness in online query processing have not been much investigated in the literature. In this short paper, we consider data quality for per-query adaptivity of the query processing, and we attempt to approach the problem of quality-aware online query processing extending our previous work in (**?**).

The remainder of this paper is organized as follows. Section 2 presents an example that illustrates the need for ensuring data quality for online query results. Section 3 states the problems to tackle for quality-aware query processing and briefly presents our proposed solutions for data quality declaration, data and metadata partitioning and quality-aware online query processing. In Section 4, we conclude and present topics for future research.

2 Revisited Example

Adapting the example of Kossmann *et al.* (**?**), Figure 1 shows the skyline of seaside hotels of Brittany (France) which are supposed to be cheap and close to the beach without considering data quality. Submitting his query, the user wants to get a *big picture* of hotels satisfying his preferences and then, choose the most "promising" hotel to make his room reservation.

In Figure 1, the connected points represent the hotels that dominate the others in terms of minimal price and distance to the beach. The underlying assumption is that the user fully trusts the quality of the data describing the hotels. Retrieved query points are located in the *2-d Data Space*.

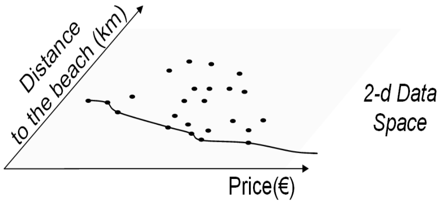


Figure 1: Classical Skyline Query Result

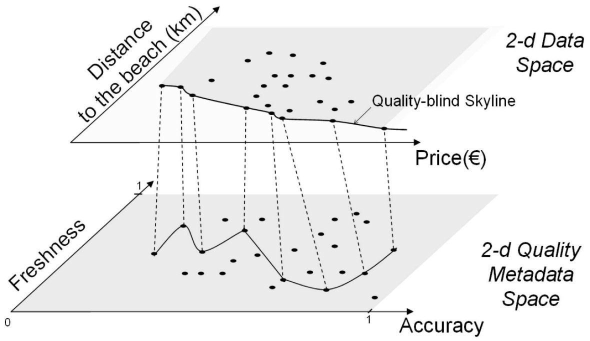


Figure 2: Quality-Blind Skyline Query Result

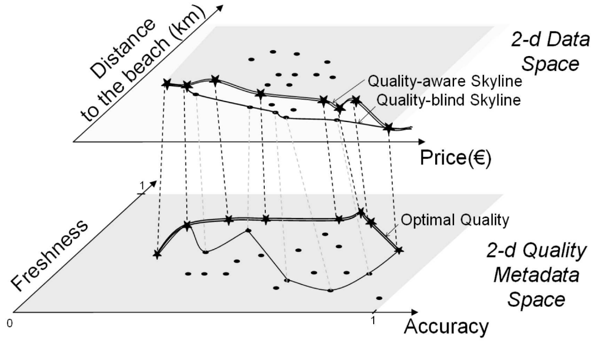


Figure 3: Quality-Aware Skyline Query Result

Consider now data quality as a "multidimensional, complex and morphing concept" (**?**). Quality metadata (such as freshness or accuracy, for example) can be associated, at a given time, to each retrieved data point with *(price, distance)* as coordinates. Figure 2 shows the *2-d Quality Metadata Space* describing data accuracy and freshness associated to each retrieved data point of the previous figure. The dotted lines joining the data points from the *2-d Data Space* to the points in the *2-d Quality Metadata Space* represent a scoring function that computes the score (as coordinate) for each dimension of quality (e.g., accuracy and freshness).

Because, the main idea of Skyline queries is to give instantaneously the user the interesting options from a potentially large set of data and, then, let the user make a decision, one might legitimately wonder if a so-called "interesting" points are meaningful when the available data are "dirty" (e.g, not accurate, not up-to-date anymore, or even worse, not complete, or not credible, etc.).

As the probability that a decision uses data increases, the needed data quality increases as well.

Actually, in this example, we observe that interesting points which are part of the initial and "quality-blind" skyline (Fig. 2) have low scores for data accuracy and freshness. In the *2-d Quality Metadata Space* of Figure 3, the optimal quality is represented as a line connecting the points ($\star$) that dominate others points in terms of maximal quality scores for data accuracy and freshness. The corresponding data points in the *2-d Data Space* (called "quality-aware" Skyline) may not be the same retrieved points of the initial "quality-blind" Skyline of hotels with minimal price and distance.

Consequently, answering online queries with data quality-awareness implies to compute interesting points (and recompute them continuously) to produce first results quickly and simultaneously check if they are also optimal in terms of data quality. This may change the initial "quality-blind" skyline of the *2-d Data Space* to produce results with optimal data quality.

3 Problems Statement and Propositions

Ensuring data quality awareness in query processing requires to propose algorithmic, computational, and technical solutions to the five following problems:

1. the definition and both offline and online computation of *generic* and *user defined* functions for measuring or predicting data quality dimensions that could be either specified or called in a declarative and flexible way, or “hard-coded” in the query processor enabling the careful quality-aware query analysis, the preparation of alternative query evaluation plans at compile time, and the selection of quality-aware query plans at run time,
2. the appropriate partition of both k-dimensional data and associated quality metadata; for the case of skyline queries, the partioning method has to enable fast and efficient nearest neighbor searches in the multidimensional data and metadata spaces,
3. the multi-objective optimization of the queries both on data dimensions and on data quality dimensions (for finding the best trade-offs between the cost, the delay of the query, and the quality of the result),
4. the adaptive query processing enabling interactive result presentation to users with possibly changing behaviours when submitting their queries and eventually their requirements or preferences in terms of data quality,
5. the transparent, reversible and explainable result presentation of “quality-augmented” queries, so that these results can be understood and accepted by the users.

In our current work, we attempt to propose several solutions for these requirements for ensuring data quality awareness in the query processing.

3.1 Declaration and Computation of Data Quality Indicators

In (**?**), we have proposed *XQuaL*, a first version of a quality-extended query language combining SQL and QML (*Quality-of-Service Modeling Language*) proposed by (**?**) and we have implemented *XQuaL* processor version 2 upon TelegraphCQ V0.2. *XQuaL* is a generic query language extension for describing and manipulating, in a flexible way, data and source quality contracts with `SFW-Qwith` queries.

A quality-extended query (`Qwith-query`) is a SFW query followed by a `Qwith` operator used to declare the required quality constraints based on the notions of quality contract types, contract instances and quality profiles. Table 1(a) presents examples of quality contract types composed of a list of ten named measurable dimensions (e.g., `dataAge` for the contract type named `Freshness`), their corresponding dimension type (noted `dim-type1`, ..., `dim-type10` for the sake of simplicity), the target of the measure that can be applied respectively on values, attribute domains, records, tables or database (noted `ON VALUE, COLUMN, RECORD, TABLE, DATABASE`), and the identifier of the measure function that computes the quality dimension indicator. For example, the function identified by `fid1` for the dimension `dataAge` computes the difference between the current date and the date of creation of each record in the database.

The creation of a contract type associated to a current database implies the execution of the identified functions (*i.e.*, the computation of the declared data quality metrics) and the creation of the contract instances that correspond to the various granularity levels of targeted data. A contract is an instance of a contract type. Table 1(b) gives examples of contract instances, respectively named `fresh`, `accurate`, `complete`, and `available` that respectively correspond to the previously declared contract types named `Freshness`, `Accuracy`, `Completeness`, and `Availability`.

The contract type `Freshness` contract has three dimensions (noted `d1`,$\cdots$, `d3` as comments in Table 1(a) and respectively named `dataAge`, `lastUpdate`, and `updateFrequency`). The type of dimension `updateFrequency` is `dim-type3` composed of a numerical value and a unit `/day` (see Table 1(b)) and it is computed by the function identified by `fid3` applied on each table of the considered database.

Similarly, the dimension `failureMasking` (`d8`) of contract type `Availability` is defined on the whole database and computed by function `fid8`. `dim-type8`, not presented for the sake of simplicity, corresponds

```
CREATE CONTRACTTYPE Freshness(
   dataAge dim-type1 ON RECORD BY FUNCTION fid1, //d1
   lastUpdate dim-type2 ON VALUE BY FUNCTION fid2,//d2
   updateFrequency dim-type3 ON TABLE BY FUNCTION fid3); //d3

CREATE CONTRACTTYPE Accuracy(
   percentageOfContradictions dim-type4 ON VALUE BY FUNCTION fid4, //d4
   NumberOfApproximateMatching dim-type5 ON VALUE BY FUNCTION fid5); //d5

CREATE CONTRACTTYPE Completeness(
   percentageOfNullValues dim-type6 ON VALUE BY FUNCTION fid6, //d6
   numberOfNullValuesPerQuery dim-type7 ON VALUE BY FUNCTION fid7) ; //d7

CREATE CONTRACTTYPE Availability(
   failureMasking dim-type8 ON DATABASE BY FUNCTION fid8, //d8
   serverFailure dim-type9 ON DATABASE BY FUNCTION fid9, //d9
   numberOfFailures dim-type10 ON DATABASE BY FUNCTION fid10) ; //d10
```

(a) Quality Contract Types

```
CREATE CONTRACT fresh Freshness(
   dataAge < 4200 seconds;
   lastUpdate == 4500 seconds;
   updateFrequency == 3 / day);

CREATE CONTRACT accurate Accuracy(
   percentageOfContradictions < 15%;
   NumberOfApproximateMatching < 1 / source);

CREATE CONTRACT complete Completeness(
   PercentageOfNullValues == 8 %;
   NumberOfNullPerQuery == 2 / query);

CREATE CONTRACT available Availability(
   failureMasking IN {noExecution, response};
   serverFailure == initialState ;
   numberOfFailures < 10 / year );
```

(b) Quality Contract Instances

Table 1: Quality Contract Declaration

to the set of possible values among `Omission`, `lostResponse`, `noExecution`, `response`. Creating quality contract types, our objective is to incorporate a set of primitives and data quality functions (e.g., `fid1`, $\cdots$, `fid10`) that can be both computed offline and recomputed or estimated at runtime.

A quality profile can be created in order to specify quality requirements and constraints by combining several instances of contract types with a particular weight. Table 2 gives an example of a profile named `quality` composed of the four previously declared contract instances (Table 1(b)) with a weighted function (`WEIGHT()`).

```
CREATE PROFILE quality(
REQUIRE(fresh, accurate, complete, available)
WEIGHT(fresh 0.4, accurate 0.3, complete 0.2, available 0.1));
```

Table 2: Quality Profile Declaration

One (or several) profile(s) may be used in the `QWITH` part of the *XQuaL* queries and applied on the attributes, tables or database involved in the query. Table 3 presents the naive "nested-loops" way to compute the revisited skyline example with data quality awareness. Price and distance values are queried from table `Hotels` and the profile named `quality` given in Table 2 is applied on the price and distance attributes with the `QWITH` operator.

```
SELECT *
FROM Hotels h
WHERE h.region='Brittany' AND NOT EXITS(
      SELECT *
      FROM Hotels h1
      WHERE h1.region='Brittany' AND
            h1.distance <= h.distance AND
            h1.price <= h.price AND
            (h1.distance < h.distance OR
             h1.price < h.price))
QWITH (PROFILE(quality) ON (price, distance));
```

Table 3: SFW-QWITH Query example

3.2 Quality Metadata and Data Partioning

As soon as a contract type is created and instanciated for the considered database, quality scores are computed for each of the targeted entities (*i.e.*, data values, columns, records, tables of the database). For the sake of simplicity, in the rest of this paper, we choose to focus on the record level among the other data granularity levels and we consider data points as vectors in a d-dimensional data space with d, the number of numerical attributes of the record. Let i ($i \in [1..n]$) representing the data points and j ($j \in [1..k]$) be the required quality profiles (e.g., combining `fresh`, `accurate`, `complete`, and `available`). Let $z_{ij} \in [min_{ij}, max_{ij}]$ be the instances of the declared contract types computed for each data point i on the jth required quality dimension.

Each data point has a scoring function $score_{ij} : [min_{ij}, max_{ij}] \rightarrow [0, 1]$ that gives the score value of the data point i assigned to the quality dimension j in the range of its acceptable values. For convenience, scores are kept in the interval $[0, 1]$.

The relative importance that the user assigns to each dimension is modeled as a weight, w_j, that gives the importance of the quality dimension j (expressed in the declaration of the profile, see Table 2). We assume the weights are normalized, *i.e.*, $\sum_{1 \leq j \leq k} w_j = 1, \forall j \in [0, 1]$. An aggregate scoring function for data point i in the k-dimensional quality space defined by $z = (z_1, \cdots, z_k)$ is defined as : $Score_i(z) = \sum_{1 \leq j \leq k} w_j \cdot score_{ij}(z_i)$.

For analytical purposes we restrict our study to an additive and monotonically increasing or decreasing value scoring function.

As an illustration based on the quality scores of several data points, we can represent a Kiviat graph on a grid of equal-distanced points (*i.e.*, points with coordinates in $[0, 1]$) such that all the vertices are both grid points and score values for each quality dimension. The circumcenter is the point where all quality dimensions are maximal and equal to 1. Figure 4 shows the Kiviat graph for five data points with their corresponding scores on ten quality dimensions (noted, `d1`, $\cdots$, `d10`). A polygon represent the quality of a data point over the specified quality contract types. Searching for the optimal quality data point (for more than three quality dimensions) correspond to find the polygon with the minimal area. Pick's theorem provides a simple formula for calculating the area A of each obtained polygon in terms of the number i of interior points located in the polygon and the number b of boundary points placed on the polygon's perimeter, as: $A = i + b/2 - 1$.

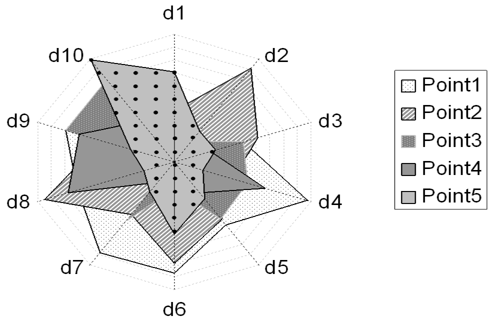
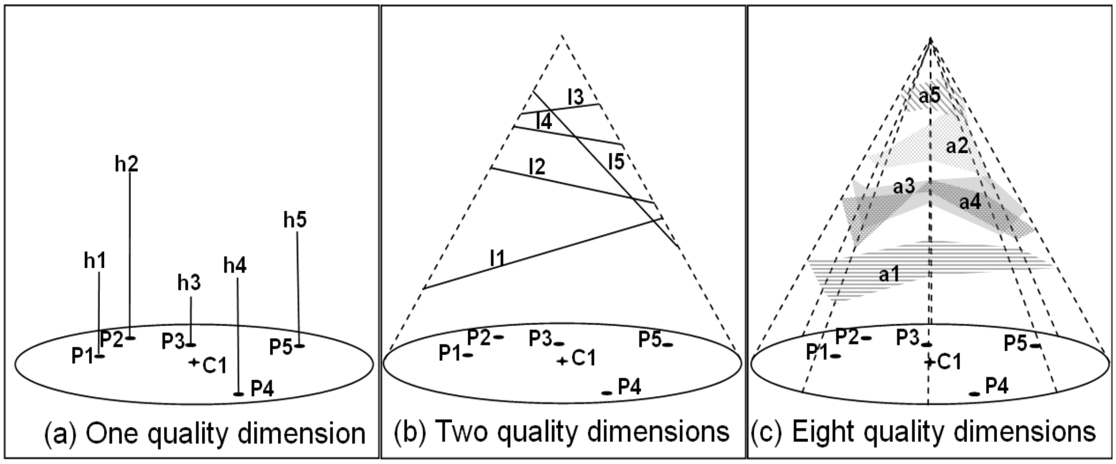


Figure 4: Kiviat Graph for Five Data Points in *10-d Data Quality Metadata Space*

Based on these considerations and faced to high-dimensional data and metadata sets, we propose a method for partitioning data and associated quality metadata. The representation, called *DQ Hyper-Pod*, consists of partitioning data that carves both the d-dimensional data space into homogeneous regions (as hyperspheres) and the k-dimensional quality metadata space into lines for $k \leq 2$ or polygon areas for $k \geq 3$. The *DQ Hyper-Pod* partition method is based on two concepts, the distance from the center of the hypersphere and the projection of the data quality scores that creates a line or a polygonal area whose center is the data point (vector) in the multidimensional data space. The lines or areas representing data quality metadata compose a pod (or envelope) upon each data hypersphere. The *DQ Hyper-Pod* partition method is defined in an Euclidean space and requires an offline phase during which data vectors are first clustered in minimum bounding hyperspheres and outliers isolated.

Consider data points in the *2-d Data Space* representing the vectors for *(price, distance)* of the Skyline of hotels of Figure 1:



(a) One quality dimension (b) Two quality dimensions (c) Eight quality dimensions

Figure 5: *DQ Hyper-Pod* Representations in *1-d* (a), *2-d* (b), and *8-d* (c) *Quality Space*

- For *1-d Quality Space*, the representation of *DQ Hyper-Pod* partitioning method is hyperspheres with orthogonal lines whose length corresponds to the quality score of each data point for the only one considered quality dimension (see Fig. 5(a)).

- For *k-d Quality Space* ($k \geq 3$), the representation of *DQ Hyper-Pod* partitioning method is right cones with their vertex above the center of their base (also the center of the hypersphere). Each cone of height h and base radius r oriented along the z-axis, with vertex pointing up, and with the base located at $z = 0$. A slant height s_j of the cone is a distance measured along a lateral face from the base to the apex. It supports one quality dimension noted j, such as the score of each data point i is located on it, as: $\forall i, score_{ij} \in [0, s_j]$ with $s_j = \sqrt{h^2 + r^2} = 1$ (see Fig. 5(c)). A polygonal area is defined for each data point joining its scores coordinates per dimension located on respectively on the slant heights of the cone.

- For *2-d Quality Space*, the representation of *DQ Hyper-Pod* partitioning method is a particular case where polygon areas are reduced to lines whose length are in $[0, r]$ and coordinates are defined by the quality scores coordinates on two opposite slant heights of the cone (see Fig. 5(b)).

In Figure 5, we consider the same hypersphere centered on $C1$ with five data points $P1, \cdots, P5$ in the three cases (a), (b), and (c) respectively for one, two or eight quality dimensions for which scores are computed simultaneously with the contract types creation.

3.3 Quality-Aware Online Queries

Back to the initial example, nearest neighbor search is applied in the context of skyline queries (**?**). First, let us recall its principle and how *DQ Hyper-Pod* partioning method can be used for data quality-awareness in the online query processing.

We assume we focus on two specific hyperspheres S_i and S_j that cluster data points in the *2-d Data Space* (see Figure 6). The minimum hypersphere bounding the data points, called S_i, is centered on C_i and its radius is r_i. $d_{\min_i}$ denotes the minimum distance between a query point q and C_i, the center of the hypersphere S_i. Based on geometrical properties of these data regions, a classical NN algorithm will first use filtering rules and discard the hyperspheres which the minimum distance to the query q is greater than the farest points of another hypersphere, as: if $d_{\min}(q, C_i) \geq d_{\max}(q, C_j)$ then disguard S_i. Then, the NN algorithm will rank the hyperspheres based on $d_{\min}$, the distance to the query point. For each hypersphere, the distances between the data points and the query point are computed and ranked. Finally, the sequential search is stopped when $d_{\min}(q, C_i) \geq d(q, nn_k)$ with nn_k the kth retrieved query points.

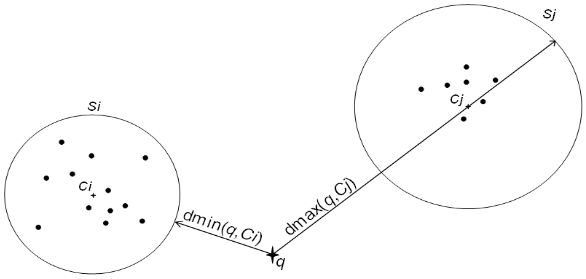


Figure 6: Quality-Blind NN Search

For ensuring data quality-awareness in this processing, we use the *DQ Hyper-Pod* partioning method and consider the quality metadata computed for each data point. Depending on the number of quality dimensions considered in the "quality-augmented" query, the NN algorithm is adapted and applied to the appropriate *DQ Hyper-Pod* representation in the different cases of one, two or more quality dimensions. Similarly to data points, the query is applied the the d-dimensional space and the k-dimensional quality space. Thus, it is a vector in $\mathbb{R}^d \times \mathbb{R}^k$. The adaptation of the NN algorithm mainly consists of two steps:

Step 1: compute the minimal distance to the query point (without considering quality, *i.e.*, $z = 0$) and rank the hyperspheres only based on the data space for retrieving the nearest neighbor,

Step 1: for each hypersphere in the list, consider the quality axis (*i.e.*, z-coordinate) and:

- in the *1-d Quality Space*, rank the data points based on the distance between each data point P_i with x-, y-, and z-coordinates, noted $P_i = (x_i, y_i, z_i)$ and the query $q = (q_x, q_y, q_z)$, as: $d(q, P_i) = \sqrt{(x_i - q_x)^2 + (y_i - q_y)^2 + (z_i - q_z)^2}$ for z_i and $q_z \in \mathbb{R}$,
- in the *2-d Quality Space*, rank the data points based on the length L_i defined by $(z_1 \cdot \frac{r}{h} + d(q, C_i), 0, z_1)$ and $(z_2 \cdot \frac{r}{h} + d(q, C_i), 0, z_2)$ with z_1 and z_2 the scores of the data point i on the two quality dimensions ($z_i \in \mathbb{R}^2$) in the hypershere centered on C_i with radius r and height h and $d(q, C_i)$ the distance between the query coordinates and the center of the hypersphere,
- in the *k-d Quality Space* with $k \geq 3$, rank the data points based on the area A_i of the polygon defined by $z = (z_1, \ldots, z_k)$ the vector of quality scores of each data point ($z_i \in \mathbb{R}^k$).

Figures 7, 8 and 9 respectively show the three case of NN search considering the quality spaces with one, two and four dimensions. In these cases, the query requires hotels with minimal price and distance to the beach and maximal data quality on the declared dimensions in the `QWITH` part of the query, that's the reason why the quality-augmented query is reduced to a single point (e.g.,$((0,0),(1,1,1,1))$ in Fig. 9), and the algorithm rank the data points based on the distance between the apex of each cone in the *DQ Hyper Pod* representation and their aggregate quality scores represented as lines or polygons.

In Figure 7, the algorithm will rank the data points inside each hypersphere based on the distance between the quality-augmented query point and the point defined by the score on the considered dimension. In Figure 8, the algorithm will rank the data points inside each hypersphere based on the length of the lines defined by the scores on the two considered dimensions.

In Figure 9, the algorithm will rank the data points inside each hypersphere based on the area of the polygon associated to each data point and defined by the scores on the various considered dimensions.

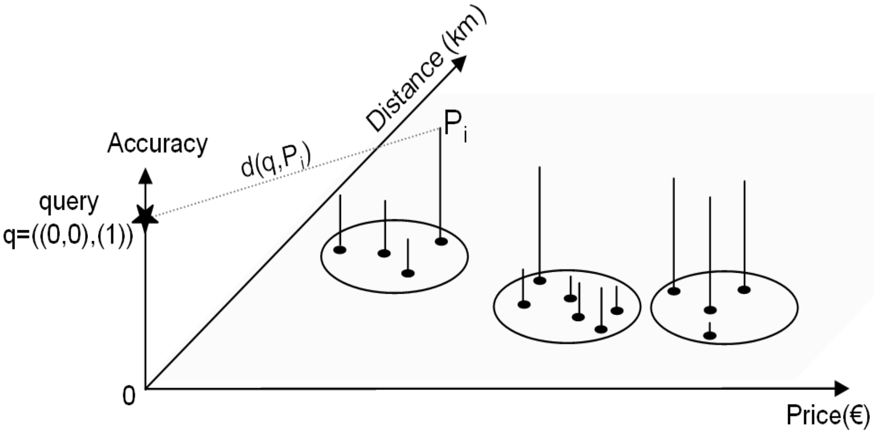


Figure 7: Quality-Aware NN search Considering One Quality Dimension

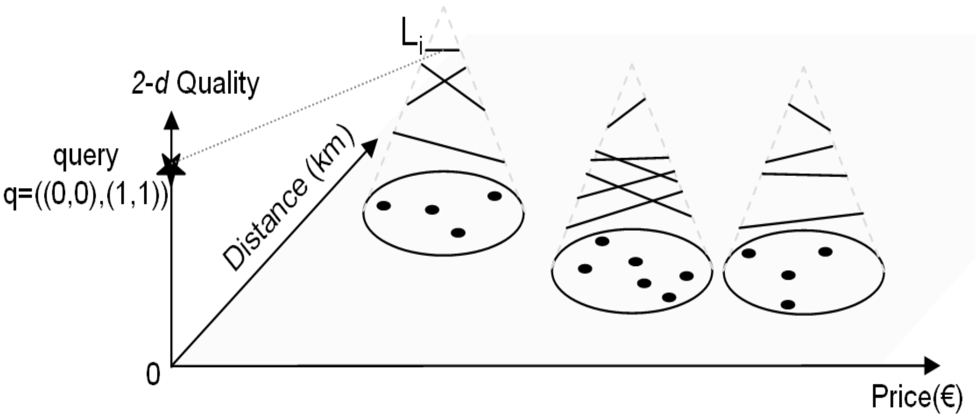


Figure 8: Quality-Aware NN search Considering Two Quality Dimensions

This technique can be easily extended to the cases where the quality scores required in the query are defined by the user and are not necessarly maximal. Hence, the `QWITH` part of the query will be evaluated and the corresponding quality scores coordinates, line length or polygon area of the query will be computed and respectively compared to the list of quality scores coordinates, line lengths or polygon areas that were pre-computed for the data points clustered in the hyperspheres of the *DQ Hyper Pod* partition.

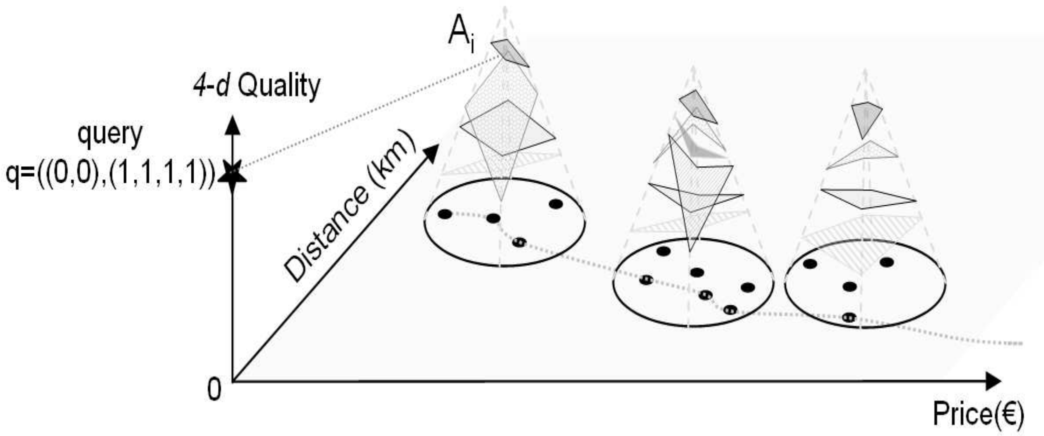


Figure 9: Quality-Aware NN search Considering Four Quality Dimensions

4 Conclusion and Future Research

As mentioned in this short paper, providing efficient access to information sources received a sustained interest since several decades but an interesting research direction in optimizing queries for single- and multi-source data management systems is the use of data quality. Few approaches have been proposed to deal with the various issues of quality-aware query processing mainly in distributed environments (HiQIQ - (**?**, **?**); ObjectGlobe - (**?**)). These issues are particularly challenging due the characteristics of the sources, including autonomy, volatility, amounts of data, large heterogeneity spectrum on *i)* data type (e.g., multimedia, XML, relational records, etc.), *ii)* on database schema, and *iii)* on the quality of data and the quality of data management services. An initial motivation is that the constraints on data quality may reflect the user's needs better in such environments. And constraints on information quality are of crucial importance for some critical applications (e.g., homeland security, business intelligence, etc.). A challenging research and development direction is to build quality-aware query processing infrastructures and this requires addressing several research issues as outlined in the following:

- *Quality of data and quality of service extended query languages*. Devise a declarative query language that targets quality of data and also quality of data management service with the advantage that the same quality-constrained query specification holds whatever underlying information is available.
- *Computation model*. In a multi-source infrastructure, the resolution of any "quality-augmented" query may involve an iterative process between the different systems. We need to devise a computation model for the interaction of the different (sub-) systems (e.g., wrapper/mediator systems, sources/data warehouse, peers, Web portals/Web services/Web providers, etc.) in order to ensure data quality awareness (through quality of data and quality of service contract negotiation, for example), not only for the query processing but also for the entire data management and processing chain.
- *Optimization model*. Performance has a prime importance in successfully deploying a quality-aware query processing infrastructure. It mainly relates to query optimization. One challenge is to define appropriate metrics to characterize and measure QoS and QoD depending on the application domain, the systems capabilities and the required performance. The different query planning strategies focus generally on finding feasible and optimal sub-goal orderings based on available bindings and supported conditions at the data sources. Proposed techniques assume a full knowledge of the query capabilities of every participating source. They rely heavily on the way that information sources are described and the objective function of the optimizer (e.g., number of sources, response time, etc.). Using the same source description

and quality description models may not always be possible across a large spectrum of data sources. The optimization of quality-aware query processing on structured data (*i.e.*, relational records) as well as on semi-structured data (XML) has to be considered. XML quality-aware query processing is still at its infancy and constitutes a very interesting trend in the near future.

- *Optimization heuristics*. In most of the real world applications, it is quite natural that "quality-augmented" query should meet a number of different and potentially conflicting quality dimensions. Optimizing a particular objective function may sacrifice optimization of another dependent and conflicting objective. An interesting perspective is the study the quality-aware query processing problem from the perspective of multi-objective optimization.

- *Quality-aware adaptive query processing*. Another interesting trend is the use of adaptive and dynamic approaches for dealing with quality-aware query optimization. This is motivated by the intrinsic dynamics of the distributed and autonomous sources where unpredictable events may occur during the execution of a query. The types of actions that are considered in these approaches fall into one of the following cases: *i)* change the query execution plans in order to privilege data quality of query results, *ii)* change the scheduling of operations in the same query execution plan or in different concurrent query plans, *iii)* introduce new operators to cater for the unpredictable events (e.g., improvement or degration of data quality), or *iv)* modify the order of inputs of binary operators. Adaptive techniques have yet to demonstrate their applicability to various real applications with large numbers of information sources. There is also a need to show how they react under heavy quality of data and quality of data management service fluctuations.

References

[1] Berti-Équille L., Quality-Adaptive Query Processing over Distributed Sources. Proc. of the 9th International Conference on Information Quality (IQ04), Cambridge, MA, USA, 2004.

[2] Braumandl R., Keidl M., Kemper A., Kossmann D., Kreutz A., Seltzsa S., Stocker K., ObjectGlobe: Ubiquitous Query Processing on the Internet. The VLDB Journal, 10(1), 2001.

[3] Dasu T., Johnson T., Exploratory Data Mining and Data Cleaning. Wiley, New York, 2003.

[4] Frølund S. and Koistinen J., QML: A Language for Quality of Service Specification. Tech. Rep. HPL-98-10, Software Technology Laboratory, Hewlett-Packard, 1998.

[5] Kossmann D., Ramsak F., Rost S., Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. Proc. of the 28th Intl. Conf. on Very Large Data Bases (VLDB), pp. 275–286, Hong Kong, China, 2002.

[6] Naumann F., Leser U., Freytag J., Quality-Driven Integration of Heterogeneous Information Systems. Proc. of the 25th Intl. Conf. on Very Large Data Bases (VLDB), pp. 447–458, Edinburgh, Scotland, 1999.

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