Provider issues in quality-constrained data provisioning

Paolo Missier and Suzanne Embury
School of Computer Science
The University of Manchester, UK
{s.embury,pmissier}@cs.manchester.ac.uk

ABSTRACT

Formal frameworks exist that allow service providers and users to negotiate the quality of a service. While these agreements usually include non-functional service properties, the quality of the information offered by a provider is neglected. Yet, in important application scenarios, notably in those based on the Service-Oriented computing paradigm, the outcome of complex workflows is directly affected by the quality of the data involved. In this paper, we propose a model for formal data quality agreements between data providers and data consumers, and analyze its feasibility by showing how a provider may take data quality constraints into account as part of its data provisioning process. Our analysis of the technical issues involved suggests that this is a complex problem in general, although satisfactory algorithmic and architectural solutions can be found under certain assumptions. To support this claim, we describe an algorithm for dealing with constraints on the completeness of a query result with respect to a reference data source, and outline an initial provider architecture for managing more general data quality constraints.

1. INTRODUCTION

An increasing number of information providers nowadays offer query services on large data sets through internet-wide published interfaces, using a variety of widely available technologies. Alongside the definition of a service interface, the stipulation of agreements regarding the quality of the service is also becoming commonplace, eg. in the form of *Service Level Agreements* [12, 2, 20]. Such agreements, however, only deal with performance issues, while the quality of the information delivered to service users is generally neglected. When compared to the more common experience of shopping for any kind of product, this situation is akin to assuming that the customers' only issue is with the opening hours of the store or the service time at checkout, with no concern for the quality of the goods – clearly an unrealistic expectation.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

We argue that data consumers are in a similar predicament: the sizable and mature body of knowledge regarding quality properties of data [18, 19, 24, 21, 9] does not translate into actionable user requirements, and yet, in simple and realistic scenarios, specific properties of data are important. Suppose, for instance, that a provider acquires copyrighted articles from publishers, and compiles independent digests and reviews of those articles, offering them for sale. While users are interested in purchasing the added-value reviews, they also want to make sure that by doing so, they are not missing the digest for any of the articles that would meet their criteria if requested directly to the publishers. For example, they want to purchase the digest for the ten most recent papers on a particular topic.

The idea at the core of our work is that users may enforce this and similar requirements by entering into a formal agreement with the added-value provider, whereby the digests produced in response to a query are guaranteed to include a sufficiently large fraction of the articles that would have been returned, had the same query been issued directly to the publisher. We refer to this property of the data as its *completeness* relative to a reference data source –in this case, the original publisher.

Thus, a completeness constraint is intended to differentiate between providers that only offer digests for a small subset of the articles that are actually available, and those that account for larger collections. Notice that, in this example, the quality of the digest itself is not part of the agreement, although it may be similarly formalized as a quality constraint, of a different type: completeness is only one of many possible properties of data for which constraints can be defined.

This simple scenario is becoming increasingly relevant in situations where (i) the data obtained from a provider has a quantifiable value to the consumer, (ii) its worthiness depends on one or more quality properties, and (ii) multiple providers may offer similar information. The combination of these factors contributes to the development of a data marketplace, whereby consumers that are interested in quality data negotiate its quality/cost trade-offs with providers. The value of quality as perceived by consumers is not necessarily only monetary. Consider for instance the case, also increasingly important, of a biologist who performs data-intensive experiments using various algorithms that operate on data obtained from public repositories (so-called *in silico* experiments). For example, the success of a gene sequence similarity analysis, consisting of matching a string sequence against a large database of known sequences, depends on

the completeness of the reference data source. Although the experiment's success criteria are normally not expressed in monetary terms, the value of using a complete reference data set is unquestionable.

The missing element that would enable data marketplaces is the ability for data providers and consumers to negotiate formal and binding agreements regarding the quality of the data. In this respect, providers seem to face the greatest challenges, as they must determine which agreement levels can be sustained, and the cost/benefit trade-offs involved. To the best of our knowledge, these issues have not been addressed, with the exception of a 1989 paper by Ballou and Tayi [5], discussed later.

This paper attempts to fill this gap. Its core contribution is a model for quality agreements, and an analysis of the issues and possible strategies available to providers that commit to such agreements. We begin by assuming that every data transaction, consisting of a query-result pair, may be subject to quality constraints. Before any such transaction may occur, providers and consumers should agree on definitions for the following elements:

- a pricing function, which associates a value to the result of any query issued by the consumer;
- a number of quality functions that formalize the notions of quality properties, and that associate a quality value to each result;
- a function of quality values for the result, that quantifies their appropriateness to the consumer. This function, not necessarily linear, is expressed as a *penalty* whose effect is to reduce the price paid for the result.

The negotiation process that leads to the specific definition of each of these elements is not relevant for our purposes, and is not considered in this paper. Before entering into such an agreement, the provider must determine its feasibility, by assessing (i) the actions required to provide data with the required quality values, and (ii) whether its data architecture supports those actions in a cost-effective way. Specifically, for each incoming query, the provider faces two problems:

1. **Detection:** it must determine to what extent a result set would incur any penalty, due to insufficient quality levels, for all the quality properties involved;
2. **Repair:** it must determine what actions are available to repair its data in order to reduce or avoid the penalties.

We base our model on the assumption that both detection and repair have a cost, forcing the provider to solve an optimisation problem involving penalties, price, and costs.

As is common with any marketing scenarios, the provider may adopt a number of different strategies for compliance. For instance, it may invest resources to proactively ensure that most of its data comply with the constraints, regardless of the specific user requests. More realistically, it may conservatively adjust the quality levels of its data, by observing the consumer's behaviour – for instance, by investing in quality only for the most popular data and accepting penalties for less frequently requested items.

Rather than focusing on any specific model, in this paper we define the provider's problem space by enumerating the factors that affect its strategies. This results in a general framework that can be used to analyze complex scenarios. The most critical factor concerns the amount of information available to ensure that the penalties assessed are *provably fair*: if we assume that the actual value of a quality function is always available both to the provider and the consumer, then (i) the fairness of the agreement can be verified, and (ii) the provider may implement an optimal provisioning strategy. For some quality properties, however, this assumption may not be realistic. For instance, evaluating the completeness of a data set relative to a reference set requires full knowledge of the latter. We model the problem of partial knowledge of quality by attaching a cost to the evaluation of quality functions; in the case of completeness, this would be the per-item cost of querying the reference source, in order to obtain a partial view of its contents. This leads to the formulation of quality estimates, which put the fairness of the agreement into question, and compromise the optimality of the provider's strategy. We propose (Section 3.1) to deal with fairness issues by introducing a third-party verification role and using a spot-check approach for ensuring limited but acceptable fairness.

As additional contribution, we present an algorithm for dealing with the specific case of data completeness constraints, first using the most favorable assumptions, including availability of quality values, and then in the more general case of partial availability. This provides an insight into the expected complexity of the general provisioning problem, and also shows a case of a property-specific algorithm that cannot be easily reused for other properties; it suggests that the generality of the solution may be limited to the detection-maintenance pattern.

As a final contribution, we describe (Section 6) a general data provider architecture that incorporates that pattern, and show how it can be implemented alongside a standard query processing engine.

Our initial investigation into the problem of provisioning data with quality constraints shows that this is a difficult one in general, although satisfactory algorithmic solutions can be found under realistic assumptions.
In the rest of the paper, quality functions are introduced in Section 2, followed by the agreement model in Section 3. The provider model is presented in Section 4, and the specific handling for completeness in Section 5. The reference architecture is discussed in Section 6. We conclude in Section 7 with our agenda for further work.

1.1 Related work

Although the specific topic of data quality agreements is new, some authors address related problems, specifically in the area of quality-aware data integration. Naumann et al. [17] assume that scores can be assigned to the data offered by multiple providers, to reflect its quality, and show that the problem of data integration in the presence of such scores results in significant extensions to known algorithms for querying data using views. We tackle a somewhat complementary problem, namely how a provider can manage its data assets in order to *obtain desirable quality scores*, which would then be passed up to an integration mediator. Similarly, a recent paper by Motro [3] shows how using utility functions may alleviate the problem of data fusion (i.e., combining different versions of the same date) in the presence of inconsistencies. Utility functions are based on quality features such

as recentness and accuracy. Using a similar perspective, the "Quality Broker" architecture presented in [15] assumes that quality features are available from several sources. The goal of the architecture, in this case, is not quality constraint satisfaction, but rather the broker-based selection of the most appropriate answer to a query, among multiple available.

A related problem is also addressed in [4], where the notion of a *data quality certificate* is presented. The purpose of the certificate is to enable reasoning about quality within the context of cooperative information systems, in order to improve the overall quality of inter-system workflows. This notion is also used, in a different form, in the quality profile model described in [13].

Underlying all of these approaches are assumptions regarding (i) a shared underlying model for quality description, and (ii) the way quality values are actually computed. While none of them seems concerned with their actual availability, in some cases, the granularity of the quality meta data is so fine – at the level of a single attribute, that the actual feasibility of computing the corresponding values may be questioned. An important but overlooked problem then becomes, to assess the robustness of these integration processes when some of the quality data is missing.

Some authors define completeness by taking into account both the size of a data source and the number of available attributes with respect to the reference source, as well as the fraction of attribute values that are non-null. Our definition of completeness only considers the size of the relation, and not the single attributes, and thus it is simpler than the *relational completeness* found in [16]. It corresponds roughly to the notion of *coverage* introduced in [11]; there, coverage is defined with respect to a universal relation, whose extension includes all tuples obtained from a number of primary data sources. Our single reference source corresponds to the universal relation.

Ballou et al. [5, 6] presented an interesting and very pertinent early attempt at linking quality properties to the effort required to provision them. There, the goal is to determine, using an integer programming model, the most effective distribution of resources that a provider can use to maintain or enhance data integrity, each with an associated cost and effectiveness. While initially assuming precise knowledge of the underlying cost, data error rate and other parameters, the model also addresses the problem of estimating some of those parameters using heuristics. How realistic the overall model is in practice, however, remains to be seen.

Finally, following the intuition that information is but another type of product, some authors have adapted established results from the practice of quality control in product manufacturing, resulting in the IP-MAP (Information Product Map) framework [23, 22, 7]. For our purposes, the merit of this work is to provide ways to *predict* quality values based on the analysis of the processes that produce those values. However, we believe that the barebone model for completeness presented in the next section would defeat the purpose of such machinery, which is best suited for complex processes with well-identified "quality weak spots."

2. QUALITY FUNCTIONS

We begin by providing functional definitions of quality properties, which we illustrate for the case of completeness, defined earlier, and of consistency, i.e., the property of a data set to conform to some validation rule.

We only consider relational data sets, i.e., extensions of a relational schema. Given two data sets D and D_r, we define the completeness of D relative to D_r as:

$$compl(D, D_r) = \frac{|D \cap D_r|}{|D_r|} \in [0, 1] \quad (1)$$

In particular, we are interested in the completeness of the result $Q(D)$ of a query:

$$compl_Q(D, D_r) = \frac{|Q(D) \cap Q(D_r)|}{|Q(D_r)|} \in [0, 1] \quad (2)$$

Intuitively, $compl_Q()$ counts the fraction of records from D_r that the user obtains by querying D rather than D_r. Recall that the reason for querying D in our examples is that it contains *added value* versions of data from D_r.

This definition is illustrated in Figure 1. Note that $compl(D, D_r)$ may be very different from $compl_Q(D, D_r)$ for some Q; even when D contains only a small subset of D_r, its completeness relative to a particular query may be high, as long as D contains most of the items that the user is requesting. In fact, the heuristics for providing completeness, presented later, are based on the provider's knowledge of the user queries; their effectiveness depends on the predictability of those queries.

Before we proceed, we must first give a precise meaning to the expression $Q(D_r)$, by clarifying the relationship between D and D_r. Let S_D and S_{D_r} be the relational schemas for D and D_r, respectively. Following the well-known "Global-as-View" pattern, described for instance by Lenzerini [14], we assume that S_D is defined as a view on S_{D_r}. Formally, this requires the definition of a mapping query mq over S_{D_r}, written as $S_D \rightsquigarrow mq(S_{D_r})$, so that any query issued to S_D can be translated into a corresponding query to S_{D_r}, through a simple process of *unfolding* of the mapping query. For example, suppose that S_{D_r} consists of two relations, $R_1(a_1, a_2, a_3)$ and $R_2(b_1, b_2, b_3)$, and that S_D consists of relation R, defined by the mapping query:

$$R(c_1, c_2, c_3) \rightsquigarrow \pi_{a_1,b_2,b_3}(\sigma_{a_3=c}(R_1 \bowtie_{a_2=b_2} R_2))$$

Then, for a query like

$$Q \equiv \pi_{c_1}(\sigma_{c_2=x}(R)),$$

the corresponding query on S_{D_r} used in the completeness definition would be

$$Q' \equiv \pi_{a_1}(\sigma_{b_2=x \wedge a_3=c}(R_1 \bowtie_{a_2=b_2} R_2))$$

To simplify the notation, in the rest of this paper we write $Q(D_r)$ instead of $Q'(D_r)$.
It is also worth mentioning that $Q(D) \cap Q(D_r) = Q(D) \cap D_r$: the use of $Q(D_r)$ only really matters in the denominator of expression (2).

As a second example of functional definition of quality property, we also define the consistency of a data item relative to a conformance rule; for instance, a rule may state that a street address in a database entry is consistent with a reference street atlas, if it can be matched uniquely against one of the reference streets in the atlas. The record matching problem has been studied extensively in the data quality literature [10, 8, 1] [1], and it is known that the evaluation of

[1]W.Winkler has made a rich collection of references to this problem available at http://csaa.byu.edu/kdd03-papers/winkler-refs.pdf.

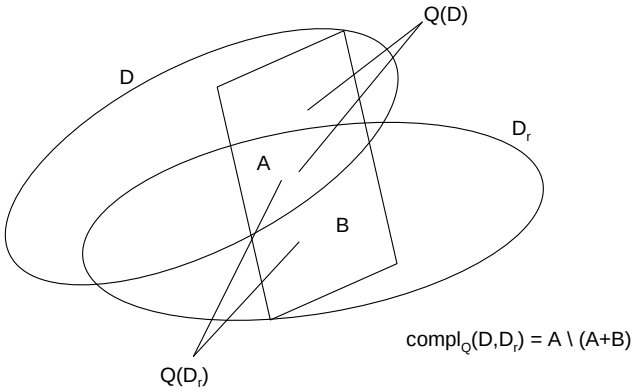

Figure 1: Completeness of a query result

this rule may incur uncertainty, accounting for the chance of false positives (an erroneus match). Thus, we assume that the rule is a function of the data and of some parameters (i.e., the reference source), and that its evaluation produces a "yes/no" result, along with a level of confidence. This function does not perform any correction or issue recommendations. In this regard, it behaves like an integrity constraint checker on a database schema.

The validation function for an item $d \in D$ with respect to D_r is

$$val(d, D_r) \in (\{true, false\}, [0, 1])$$

where the second component of the value is the confidence in the outcome. Given a user-defined threshold c_0 for the confidence, let the set of *acceptable* items relative to *val* and c_0 be

$$acc(val, D, D_r, c_0) = \{d \in D | val(d, D_r) = (true, c) \wedge c \geq c_0)\}$$

A simple definition for the consistency of D relative to D_r, given c_0, is the proportion of acceptable items in D. For $Q(D)$ this is written as:

$$cons_Q(val, D, D_r, c_0) = \frac{|acc(val, Q(D), D_r, c_0)|}{|Q(D)|} \in [0, 1]$$

This simple definition illustrates the general idea of consistently defining normalized quality functions, using user-specified parameters such as the threshold c_0.

3. AGREEMENT MODEL

Agreements are based on a simple penalty/reward model, whereby the consumer and the provider agree on formally defined quality constraints for the data provisioned on a query-by-query basis, the provider may charge fees in return for data, and the consumer may assess penalties when the quality constraints are violated.

Rather than defining discrete constraints, i.e., of the form $compl_Q(D, D_r) > compl_{min}$, we instead allow for a more general formulation, by associating penalty functions of arbitrary shapes to quality functions. When applied to a base price for a query result, they reduce the actual fee paid, in a way that is proportional to the perceived importance of the specific property.

For query Q on D, the agreement includes the following elements:

- a set of normalized quality functions of the form

  $$qf(D', \mathcal{P}) \in [0, 1]$$

  for any $D' \subseteq D$, where 1 is the best quality achievable. $\mathcal{P}$ indicates a property-specific set of additional parameters, eg. D_r, c_0. In particular, we are interested in computing the quality associated to a query result, $qf(Q(D), \mathcal{P})$;

- a base pricing function $price_b(D')$ for any $D' \subseteq D$. Again, in practice we are interested in computing $price_b(Q(D))$;

- a penalty function $pen_{qf}(D', \mathcal{P}) \in [0, 1]$, which introduces a bias on the base price, by mapping the values of the normalized quality function qf applied to D', onto a penalty factor.

The actual price paid by the user for $Q(D)$ is then

$$price(Q(D), qf(), \mathcal{P}) = price_b(Q(D)) \times (1 - pen_{qf}(Q(D), \mathcal{P}))).$$

Note that discrete quality constraints can be expressed simply by defining binary penalty functions. For example, the following constraint makes the result set worthless if the completeness falls below a threshold $compl_{min}$:

$$pen_{compl}(D', D_r) = \begin{cases} 0 & \text{if } compl(D', D_r) > compl_{min} \\ 1 & \text{otherwise} \end{cases}$$

Normally, however, the penalty will be proportional to the quality level, i.e., the function is monotone decreasing in the value of $qf()$. Although no assumptions on the shape of penalty and pricing functions need to be made, this additional information helps in reducing the complexity of the provider algorithms for managing quality compliance. The baseline algorithm presented in Section 5.1, shows a worst case scenario, in which no assumption is made regarding the shapes of these functions.

Note also, that using normalized quality functions makes it straightforward to extend this pricing scheme to multiple quality properties, i.e., by combining multiple quality and penalty functions:

$$price(Q(D), \{qf_i()\}, \{\mathcal{P}_i\}) = price_b(Q(D)) \cdot \prod_i (1 - pen_{qf_i}(Q(D), \mathcal{P}_i)))$$

Functions $\{qf()_i\}$, $price_b()$ and $\{pen_{qf_i}()\}$ are all defined as part of a negotiation process, whose details are not relevant for our purposes. Once the agreement is in place, it is enforced as follows:

- for every incoming query Q, the provider computes $D' = Q(D)$;
- for every $qf_i()$ that is subject to a penalty, compute $q_i = qf_i(D', \mathcal{P}_i)$ and $p_i = pen_{qf_i}(q_i)$;
- compute the final price $price_b(D') \cdot \prod_i (1 - p_i)$.

3.1 Fairness of penalty assessment

As anticipated, the fairness of the penalty assessment depends upon the value of $qf()$ being available. In our example, this corresponds to having a catalog of all available articles, which can be queried (this is $Q(D_r)$), regardless of how many of those articles have received a review. Similarly, in the biology database case, a public source for the primary data may be available free of charge. While these are reasonable assumptions, in general we also need to consider the cost of computing $qf()$. When the provider incurs this *monitoring cost* (see Section 4), it may need to compute an estimator $\hat{qf}()$ of the actual $qf()$, balancing its precision with the cost of limited monitoring.
The idea is then to let providers self-assess their penalties, and to adopt the simple but widespread view that a third-party verification authority is in charge of assessing the correctness of penalties. We assume that this authority also incurs a monitoring cost. For completeness, this results in the following scenario:

- the provider defines its own estimators $\hat{qf}()$ for $qf()$, based on some probabilistic model and by sampling using a limited number of $Q(D_r)$ queries, determined by its budget[2];
- the authority has its own strategy for verification, which relies on spot-checks performed at some time intervals, again by querying $Q(D_r)$;
- the provider may negotiate a tolerance $\delta \in [0, 1]$ as part of the agreement, which limits its liability vs the customer in case of imprecise estimates;
- whenever the authority determines the actual value for $qf()$, the percentage estimation error $\hat{e} = \frac{\hat{qf}() - qf()}{qf()}$ is computed, and the provider incurs a fine that is proportional to $\hat{e} - \delta$, whenever $\hat{e} > \delta$.

As a result, the provider's chance of getting away with an incorrect estimate (and hence, a reduced penalty) depends on the authority's budget and ability to monitor effectively.

In this scheme, the consumer relies on the authority for control. In case of dispute of past transactions, the authority is obliged to perform a check on a past quality value, which may require enabling infrastructure. For completeness, this amounts to querying a past state of the D_r database – which is feasible if D_r is a standard transactional DBMS with logging capabilities.

[2]We are implicitly assuming, for the purpose of the example, that the monitoring cost in this case is proportional to the number of items retrieved from D_r.

4. PROVIDER COMPLIANCE MODEL

In this section we analyze the issues associated to enforcing an agreement, from the provider's perspective. The quality-constrained data provisioning problem can be described according to a simple "monitor-assess-repair" reactive model:

monitor: Firstly, the provider must compute the value of quality functions every time it receives a query. Since some of the function parameters may not be available, i.e., $Q(D_r)$, they must be estimated;

assess: Secondly, the provider must estimate the penalty associated with the result set for the query;

repair: Thirdly, it must determine the most cost-effective repair actions to be executed in order to move the state of its data set towards compliance.

With reference to completeness and consistency, the model is instantiated as follows.

Detection. We denote with $\overline{D}_Q$ the quantity we wish to monitor. For completeness, this quantity is

$$\overline{D}_Q, compl = Q(D_r) \setminus Q(D)$$

This set contains all the items that the user would have obtained by issuing Q to D_r, but are instead missing from $Q(D)$. These are therefore the items responsible for the penalties incurred when returning $Q(D)$. For consistency, $\overline{D}_Q$ is the set of items that were expected to be consistent, but are not:

$$\overline{D}_Q, cons = Q(D) \setminus acc(val, Q(D), D_r, c_0)$$

Repair. For completeness, the only repair procedure consists in obtaining new data items from D_r. For consistency, the procedure may perform validation on items whose consistency is unknown, and apply algorithms to enforce consistency (for instance, by correcting data or obtaining new versions from a third party source).

Costing. The third step is the choice of a provider cost model, which includes a *monitoring* component $cost_m(Q, D)$, a *repair* component $cost_r(D)$, and the *value-adding* component $cost_{va}(d)$ of expending local resources in order to prepare any $d \in D$ for delivery.[3] For completeness, the first two correspond to the cost of computing $\overline{D}_Q, compl$ and the cost of obtaining a new item d from D_r, respectively.

Compliance strategies. Next, the provider must identify a strategy for activating repair procedures given the price and penalty information from the agreement, the observed violations, the cost model, and a goal. An obvious general provider goal is to avoid penalties that erode profit, by incurring the minimal repair cost. For completeness, this translates into the strategy of obtaining the smallest set of items from D_r, which restores the required completeness levels. As noted, D may be a small subset of D_r, however if it contains the items that are most likely to be requested in the future, these may be sufficient to satisfy the constraints for most of the user queries. Therefore, a sensible approach is to use the history of past queries to estimate the likelihood of any item in D_r being requested in the future. Estimating future request probability is clearly more effective than a simpler greedy strategy, which would acquire only the items

[3]This could for example be the cost of producing a review for a new article.

that are missing from the current query, some of which will not be requested again.
Regardless on the specific choice of estimator, the precision (i.e., confidence level) associated to the estimate depends on the length of query history: for a new agreement or a new user, the provider will be able to do little more than repairing based on the current query. Various statistical models can be developed to estimate such probability, and it is not the purpose of this work to survey them. Simple estimators include the frequency of past requests, which assume that the past popularity of an item is an indicator of future interest; a mobile average on a limited time window, which attempts to track the changes in interest; or an estimator based on the hypothesis that the occurrence of a request has a known distribution. We note in passing the similarity between the problem of predicting the request of items that are obtained from reference sources, and the problem of defining cache replacement algorithms: for properties like completeness, similar estimators may be applicable.

4.1 Factors that affect compliance strategies

A number of factors complicate the choice and implementation of a strategy:

- **Instant repair:** is it feasible for the provider to obtain new items from D_r, and use them to repair the current result set? When this is not possible, current penalties are inevitable, and the repair strategy may only focus on avoiding future penalties.

- **Completeness of detection:** has the provider complete knowledge of the quality indicators? The provider must balance the precision of the $\overline{D}_Q$ estimate, with the monitoring cost $cost_m()$ of computing it, and the chance that the verification authority will claim irregularities.

- **Number of quality properties** that appear in a single agreement, or in multiple agreements: the potential interaction between constraints on different properties may complicate the strategy. Consider for instance *currency*, the property of a data value of being correct at a given time[4]. Currency and completeness are not independent, because if we assume that all items in D_r are current, then obtaining a new item from D_r in order to restore completeness, has also the effect of improving currency. On the other hand, given a budget for acquiring new items, there may be a contention between different quality constraints that depend on those items for their satisfaction, breaking the isolation of single-property strategies.

- **Options available for detection and repair:** while for completeness the only repair option is to obtain items from a reference source, multiple such sources may be available, possibly at different costs. Also, other properties may present richer options: validating an item for consistency may involve requesting a correction from a reference source, or performing manual inspection on the item.

[4]The data for an address that changed recently may have been correct before the change occurred, but it has since become obsolete, or non-current, until it is updated.

- **Notification of updates to a reference source:** for properties whose repair actions depend on a reference source, it matters whether the provider is notified of any update to the source. For instance, if completeness is estimated by periodically sampling D_r, then the estimate is affected by the frequency of updates to D_r.

- **Shape of cost and price functions:** the various cost and price functions listed earlier may depend on the particular choice of item, or may be defined as a function of the size of the result.

5. PROVISIONING WITH COMPLETENESS

In order to provide a concrete example of provisioning with quality, we now illustrate an algorithm for completeness, based on the detection-repair model. With respect to the complicating factors listed above, the assumptions for the algorithm are as follows: instant repair is possible; only one quality property appears (completeness), and the only repair option is to obtain new items from the reference source; there is no notification of updates to the reference; and finally, the price function depends only on the size of the query result.

We first present a baseline algorithm that assumes that the provider has complete and free knowledge of the quality indicators, and then propose a generalization that does not require this assumption.

5.1 Baseline algorithm

The approach is based on the definition of a utility function for a set $D' \subseteq D_r \setminus D$ of currently missing items, and the formulation of a corresponding optimization problem, that can be solved using heuristics. For completeness, the function takes into account the provider cost model and the penalty functions:

$$\begin{aligned} U(D', Q, D, \mathcal{P}) \;=\; & price_b(Q(D) \cup (D' \cap Q(D_r))) \\ & \times (1 - pen_{compl}(Q(D) \cup (D' \cap Q(D_r)), \mathcal{P})) \\ & - cost_r(D') - cost_{av}(D') \end{aligned}$$

In practice, U describes the effect of purchasing set D':

- the base price is increased due to the new items in D'. Notice that there is no guarantee that the algorithm will only purchase items that are missing from the current result. In fact, the heuristic presented later makes a less greedy selection, hoping to improve *future* compliance. Hence, only the items in $D' \cap Q(D_r)$ contribute to the immediate extra reward.

- The penalty is reduced correspondingly (the same observation applies).

- The cost incurred is due to purchasing and adding value to D'.

The optimization problem is designed to limit the risk of purchasing items that may not be needed in the future: we are seeking the subset of $D_r \setminus D$ that maximizes the ratio of utility-to-size:

$$\max_{D' \in D_r \setminus D} \frac{U(D', Q, D, \mathcal{P})}{|D'|}$$

Normalizing by size avoids the effect of an indefinitely increasing utility, which would result in purchasing the largest possible D'. Note that the problem is defined on the entire set of missing items, rather than only on $\overline{D}_Q$, and that it must be solved when Q is computed.

Since we are not making any assumptions on the shape of function U, or of any of its components, a brute-force algorithm that enumerates all possible D' has exponential complexity in $|D_r \setminus D|$. To reduce the complexity, we apply the strategy mentioned in Section 4, using the history of past user queries to estimate the likelihood of a missing item to be requested in the future.

Algorithm 1. For each Q, the provider maintains a count of the frequency of requests of each item $d_i \in \overline{D}_Q$, and requires an estimate of the likelihood of a future request for each d. Since this involves updating the frequency of the items that are actually requested, complete knowledge of D_r is not required. The algorithm starts from an empty set D', and incrementally adds to it in order of estimated likelihood, recording the value of U at each step. In this way, at step i only the most promising of the $\binom{|D_r \setminus D|}{i}$ potential sets is considered. □

Some comments are in order:

- From the definition of the utility function, we note that its components depend not only on the number of items, but also on their choice. This is because we allow the selection of D' to range on the entire set $D_r \setminus D$, rather than only on $\overline{D}_Q$, hence some of the selected items may not reduce the immediate penalty.
- The order defined on the missing items is partial. For instance, after the first query, the best set contains a random selection (of optimal size) of items from $\overline{D}_Q$, because all such items have the same frequency of occurrence. We assume that the items in $\overline{D}_Q$ are preferred over others of the same rank.
- It is worth considering the effect of a *locality principle* on this heuristic, which states that the history of past queries is indeed a good predictor for future queries. Consider what happens when queries are highly localized and an occasional odd query arrives, requesting items never mentioned before. Since these items have a low frequency, they are ranked low relative to others that have been requested in the past multiple times, but are still missing. In this case, the algorithm does not try to repair the current query (which will therefore result in a penalty), but rather it will purchase additional popular items, increasing the expected reward for future queries.
- As noted earlier, initially the limited history of past requests makes the estimators unreliable, yielding items that may in fact never be used again in the future. This confirms the intuition that this agreement model makes frequent and regular consumers more appealing than occasional ones, and suggests that quality agreements are best suited for long-term consumer-provider relationships.

We conclude by noting the effect of updates and inserts into D_r. In this version of the algorithm, even if the provider is not informed of these events (for instance, through some event notification infrastructure), they do not pose problems.

When an update comes to D_r, then D clearly holds a stale copy, of which it is not aware. However, unless there is an explicit currency constraint, this has no consequences on completeness! – this is in fact a case for handling completeness and currency together. When a new insert occurs in D_r, according to our algorithm it may be revealed only through queries of the form $Q(D_r)$. Items that appear in D_r but are never requested, are simply ignored. Items that start being requested after they have been inserted, are handled in the same way as all others.

5.2 Ranking of missing data using query predicates

Given a query of the form $Q = \sigma_p(R)$, our baseline algorithm relies on the *extension* D_r for computing completeness (detection), and for selecting the most promising items to purchase (repair), assuming $Q(D_r)$ known, by simply enumerating the missing items. The algorithm described in this section addresses the problem of performing detection and repair when $Q(D_r)$ is not available.

The idea is to consider only the *conditions* stated in the user queries, and those used by the provider to obtain new items from D_r. We rely on two observations: firstly, that the most popular data are represented at the *intensional* level by the history of user queries; and secondly, that although $Q(D_r)$ is not immediately available, within the limited scope of a specific user query we may still provide a good estimate for the completeness $compl_Q(D, D_r)$, and also determine the conditions corresponding to the most popular items in D_r, for repair.

The algorithm is based on the definitions of *request profiles* and *completeness maps*. Similar in spirit to ordinary database profiles used by relational query optimizers, a request profile records the level of interest for specific data items, and is computed progressively from a history of user queries. The main difference is that, while in ordinary profiles the data points in the histograms are attribute values, in this case they are *query predicates*.

Whereas a request profile records the demand for data, a completeness map represents the available data set, as described by the set of queries issued by the provider to D_r. Intuitively, knowledge of the user requests to the provider and of the provider requests to its suppliers is sufficient to rank the data that the provider is missing, according to the user preferences.

Let $\mathcal{Q} = Q_1, Q_2, \ldots, Q_m$ and $\mathcal{Q}' = Q'_1, Q'_2, \ldots, Q'_n$ be the history of user and provider queries, respectively. We may restrict our attention to select-queries only, of the form $Q = \sigma_p(R)$, ignoring projections; having defined completeness at the granularity of the entire data item, a distinction based on projected attributes is unnecessary. Furthermore, we make the simplifying assumption that selection predicates are conjunctions of elementary conditions that are either (i) expressions involving relational operators *relop* $(=, \leq, \geq)$ on ordered domains, of the form x *relop* c, or (ii) set membership expressions on enumerated sets, i.e., $x \in \{c_1, \ldots, c_n\}$.

Given a relation $R(A_1, \ldots, A_l)$, these definitions are formalized as follows.

DEFINITION 1. *(Request profile)*

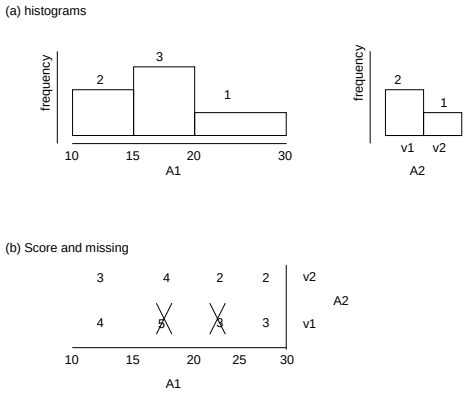

Figure 2: Construction of request profiles

Given set $P_i = \{p_{i1}, \ldots, p_{in_i}\}$ of logically disjoint predicate expressions p_{ij} on A_i, a request profile is a set of l histograms

$$freq_i : P_i \rightarrow \mathcal{N},$$

one for each A_i. Each $freq_i$ maps P_i onto its frequency of occurrence as observed in the query history $\mathcal{Q}$.

For example, suppose that D is described by the single table $R(A_1, A_2)$, where A_1 ranges over the positive integers, and A_2 ranges over the finite set $\{v_1, \ldots, v_n\}$. Let the predicates found in the history of queries be:
$p_1 = (A_1 \in [10, 20])$
$p_2 = (A_1 \in [10, 20] \wedge A_2 = v_1)$
$p_3 = (A_1 \in [15, 30] \wedge A_2 = v_2)$
$p_4 = (A_2 \in \{v_1, v_2\})$
We write $[10, 20]$ as a shorthand for $A_1 \in [10, 20]$. The histograms are constructed as follows:

1. Q_1 carries predicate $[10, 20]$, so $freq_1([10, 20]) = 1$.
2. After Q_2, $freq_1([10, 20]) = 2$ and $freq_2(v_1) = 1$.
3. Q_3 causes the $[10, 20]$ interval to be *refined* into the adjacent disjoint intervals $[10, 15]$, $[15, 20]$ and $[20, 30]$, associating a frequency to each:
   $freq_1([10, 15]) = 2$,
   $freq_1([15, 20]) = 3$,
   $freq_1([20, 30]) = 1$
   (partitioning two overlapping intervals into disjoint intervals can be accomplished easily). Also, now $freq_2(v_1) = 2$ and $freq_2(v_2) = 1$.

The resulting histograms are illustrated in Figure 2 (a). Notice that these histograms are independent of each other: for simplicity, we do not account for the co-occurence of some of the predicates within the same query. Also, by construction the histograms only include the predicates that appear in the queries, rather than all possible combinations for the value set of each attribute.

Definition 2. *(Space of predicates)*
Given the sets $\{P_1, \ldots, P_l\}$ of predicate expressions for a request profile, the space of predicates $\mathcal{P}$ is the set of all vectors of the form $\mathbf{p} = (x_1, \ldots, x_l)$, with $x_k \in P_i$.

In the example, $\mathcal{P}$ includes $([10, 15], v_1)$, $([10, 15], v_2)$, $([15, 20], v_1)$ and so forth. We describe the popularity of a combination $\mathbf{p} \in \mathcal{P}$ of predicates using a syntectic score value:

$$score(\mathbf{p}) = \sum_{i:1..l} freq_i(p[i])$$

i.e., $score([15, 20], v_1) = 5$, $score([20, 30], v_2) = 2$, etc.

This score, however, is oblivious of the data from D_r that has already been purchased. Thus, it is complemented by the partial information on the completeness D relative to D_r:

Definition 3. *(Completeness map)*
Given $\mathbf{p} = (x_1, \ldots, x_l) \in \mathcal{P}$, let $p = x_1 \wedge \cdots \wedge x_l$ be a predicate. A completeness map is described by boolean function

$$missing(p) \in \{true, false\}$$

defined on the space of predicates, such that

$$missing(p) = true \text{ iff } Q_p(D_r) \subseteq D.$$

Assuming $missing(p) = true$ for all p initially, the map is updated using the history $\mathcal{Q}'$ of data purchases. For instance, let $p'_1 = (A_1 \in [15, 25] \wedge A_2 = v_1)$ be the predicate for Q'_1. As shown in Figure 2 (b), first p'_1 is used to further refine the histogram for A_1, adding the interval boundary 25. The corresponding *score* table is updated as a consequence. Splitting $[15, 25]$ into $[15, 20]$ and $[20, 25]$ aligns this interval with the existing histograms. Then, we set $missing([15, 20], v_1) = missing([20, 25], v_1) = 0$. In practice, we mark selected points in the space of predicates, which represent conjunctions that have already been used to purchase new data.

The rank of a point $\mathbf{p}$ representing missing data is simply the product

$$rank(\mathbf{p}) = score(\mathbf{p}) \ . \ missing(\mathbf{p})$$

This ranking provides a preference only among the predicates that describe popular items that are still missing (the others have rank 0), and replaces the simpler repair criterion used in the baseline version of our algorithm. In the example, the combination $([10, 15], v_1)$ is the most popular, since $([15, 20], v_1)$ is not missing.
It is worth mentioning that the operation of histogram refinement that may follow each user query, also requires reevaluating the *missing* function. This is simple, however, since each new sub-interval inherits the *missing* values found in the parent interval (this is left to intuition).

Algorithm INTENSIONAL DATA RANKING
Given relation $R(A_1, \ldots, A_l)$:

For user query $Q = \sigma_p(R)$:
begin
  for each A_i **do** {
    Let p_i be the conjunction of terms from p on A_i;
    affectedPoints = $refine(P_i, p_i)$;
    $updateHistogram(freq_i(p_i))$;
  }
  for each **p** $\in$ affectedPoints **do** $updateScore(\mathbf{p})$;
  for each **p** $\in \mathcal{P}$ **do** $rank(\mathbf{p}) = score(\mathbf{p}) \times missing(\mathbf{p})$;
end

For provider query $Q' = \sigma_{p'}(R)$ issued to D_r:
begin
  for each A_i **do** {
    Let p'_i be the conjunction of terms from p' on A_i;
    affectedPoints = $refine(P_i, p'_i)$;
    for each **p** $\in$ affectedPoints **do** $updateScore(\mathbf{p})$;
    newPoints = $computeNewPoints(p'_i)$;
  for each **p** $\in$ newPoints **do** $missing(\mathbf{p}) = false$;
end

Figure 3: Ranking algorithm

The ranking algorithm is summarized in Figure 3. Function $refine()$ increases the number of intervals in the histogram, and returns the points in the space of predicates that are affected by this operation. For these points, the score is updated prior to computing their rank. Upon issuing Q' to D_r, the provider must additionally compute the new points in the space of predicates corresponding to the query predicate, as shown earlier in the example, and reset their missing flag.

Tables *score* and *missing* can also be used as a basis to provide various estimates of completeness for a query $Q = \sigma_p$:

$$compl_{\sigma_p}(D, D_r) = \frac{|\sigma_p(D) \cap \sigma_p(D_r)|}{|\sigma_p(D_r)|}$$

For example, given predicate $p = [10, 20]$ with the situation illustrated in Figure 2, one may view intervals as discrete elements, regardless of their width, and observe that p corresponds to the "slice" of the *score* table that includes 4 predicate combinations. Since only one of these is not missing, one may estimate $compl_p(D, D_r) = 0.25$. Alternatively, the width of each of the intervals involved or other weight factors can be taken into account, yielding different estimates.

6. REFERENCE ARCHITECTURE

The architecture sketched in Figure 4 consists of a quality management module that contains the key components described in the previous sections. When a user query comes through the client service interface, it is processed as usual; before the result is returned, it is intercepted and passed to the quality management module, and query post-processing occurs. Using the interceptor pattern ensures that no changes are required to the query processor.

With reference to completeness, query post-processing proceeds as follows. The detection component is in charge of monitoring the completeness indicator and of computing the quality value, estimating the penalty associated with the query result. As explained in the previous section, this is done by querying the profile manager, which controls the completeness maps. The user query is also used by the profile manager to update the request profiles.

The strategy manager then uses this information to compute the utility function and to setup the optimization problem. Again, the profile manager is a critical component, in that it provides the ranked predicates that correspond to the most interesting new items. At this point, the repair actions consist of one or more queries to D_r, issued by the repair component using the pull interface of the gateway.

If the "instant repair" option is available, described in Section 4.1, then the new data items are prepared for immediate delivery and added to the original result set. Additionally, the queries are passed to the profile manager, which updates the completeness map.

If a push interface is active, then any update to D_r is propagated not only to D, but also to the profile manager, which may use it to update the completeness estimates. At the end of post-processing the final price is computed, taking self-assessed penalties into account, and the result is returned to the user. In the figure, an agreement interface is also shown as part of the quality management module, to indicate the channel used for agreement negotiation, which results in the configuration of the module components.

7. FURTHER WORK AND CONCLUSIONS

We have presented a simple model for the definition of formal agreements between data providers and consumers regarding the quality levels of data, illustrating the provider's problem space and showing an algorithm for dealing with the completeness property as a special case. Finally, we have described a reference provider architecture for enforcing quality agreements, that is respectful of the existing query processing architecture.

This work is at its initial stages and can be extended in many directions, adding elements that may affect our initial assessment of the agreement model and of the providers' strategies. Firstly, we believe that an experimental evaluation of the presented approaches for completeness, and a prototype implementation of the architecture, may provide an insight into their practicality. Secondly, the overall framework and architecture must be tested by considering additional properties: are there suitable architectural patterns for dealing with constraints on multiple quality properties, and how would the provider define strategies that involve interplay between properties, i.e., between completeness and currency? In the same vein, dealing with multiple agreements with overlapping constraints may intuitively make the provider's strategy more cost-effective, in ways that need to be investigated.

8. REFERENCES

[1] A.Borthwick, M.Buechi, and A.Goldberg. Key concepts in the choicemaker 2 record matching system. In *Procs. First Workshop on Data Cleaning, Record Linkage, and Object Consolidation, in conjunction with KDD 2003*, Washington, DC, July 2003.

[2] A.Dan, D.Davis, R.Kearney, A.Keller, R.King, D.Klueber, H.Ludwig, M.Polan, M.Spreitzer, and

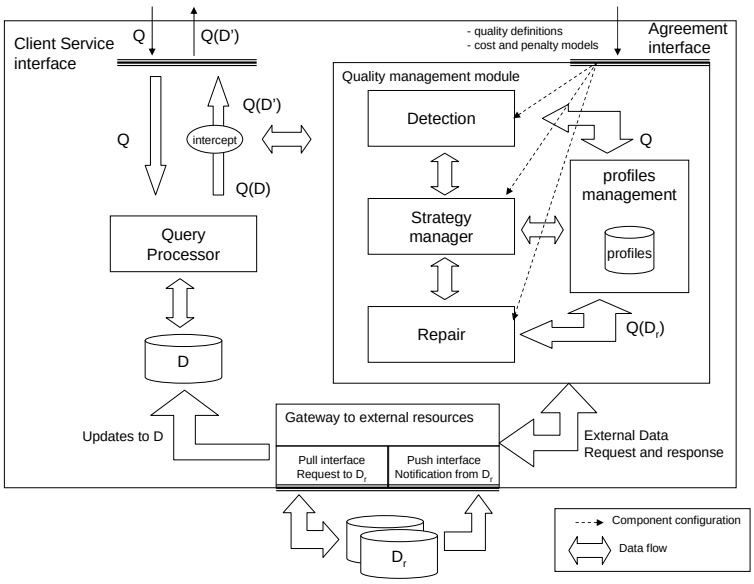

Figure 4: Reference architecture

A.Youssef. Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, 43(1), 2004.

[3] A.Motro, P.Anokhin, and A.C. Acar. Utility-based resolution of data inconsistencies. In Felix Naumann and Monica Scannapieco, editors, *International Workshop on Information Quality in Information Systems 2004 (IQIS'04)*, Paris, France, June 2004. ACM.

[4] C. Cappiello, C. Francalanci, P. Missier, B. Pernici, P. Plebani, M. Scannapieco, and A. Virgillito. Presentation of metadata and of the quality certificate. Deliverable dl2, The DaQuinCis project, 2003.

[5] D.Ballou and G.K.Tayi. Methodology for allocating resources for data quality enhancement. In *Communications of the ACM*, volume 32. ACM, March 1989.

[6] D.Ballou and H.Pazer. Designing information systems to optimize the accuracy-timeliness tradeoff. *Information Systems research*, 6(1), 1995.

[7] D.Ballou, R.Wang, H.Pazer, and G.K.Tayi. Modelling information manufacturing systems to determine information product quality. *Journal of Management Sciences*, 44(4), April 1998.

[8] M.G. Elfeky, A.K. Elmagarmid, and V.S. Verykios. Tailor: a record linkage tool box. In *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, San Jose, CA, Feb. 2002. IEEE Computer Society.

[9] L. P. English. *Improving data warehouse and business information quality: methods for reducing costs and increasing profits*. John Wiley & Sons, 1 edition, March 1999. ISBN: 0471253839.

[10] I.P. Fellegi and A.B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64, 1969.

[11] F.Naumann, J.C.Freytag, and U.Leser. Completeness of integrated information sources. *Information Systems*, 29(7):583–615, 2004.

[12] S. Kalepu, S. Krishnaswarmy, and S.W. Loke. Verity: a qos metric for selecting web services and providers. In *Proceedings of 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03)*. IEEE Computer Society Press, 2004.

[13] P. Missier and C. Batini. A multidimensional model for information quality in cooperative information systems. In M. Helfert M. Eppler, editor, *Proceedings of the Eight International Conference on Information Quality (ICIQ-03)*, 2003.

[14] M.Lenzerini. Data integration: A theoretical perspective. In *Principles Of Database Systems*, pages 233–246, 2002.

[15] M.Scannapieco, A.Virgillito, C.Marchetti, M.Mecella, and R.Baldoni. The architecture: a platform for exchanging and improving data quality in cooperative information systems. *Inf. Syst.*, 29(7):551–582, 2004.

[16] M.Scannapieco and C.Batini. Completeness in the relational model: a comprehensive framework. In *Procs. 9th International Conference on Information Quality, ICIQ 2004, Cambridge, Ma*, 2004.

[17] F. Naumann, U.Leser, and J.C.Freytag. Quality-driven integration of heterogenous information systems. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, pages 447–458, Edinburgh, Scotland, UK, September 1999. Morgan Kaufmann.

[18] T.C. Redman. *Data quality for the information age*. Artech House, 1996.

[19] R.Y.Wang, M.Ziad, and Y.W.Lee. *Data quality*. Advances in Database Systems. Kluwer Academic Publishers, 2001.

[20] J. Skene, D. D.Lamanna, and W. Emmerich. Precise service level agreements. In *Proceedings of 26th International Conference on Software Engineering (ICSE'04)*. IEEE Computer Society Press, 2004.

[21] Y. Wand and R.Wang. Anchoring data quality

dimensions in ontological foundations. *Communications of the ACM*, 39(11), 1996.

[22] R. Wang. A product perspective on total data quality management. *Communications of the ACM*, 41(2), February 1998.

[23] R. Y. Wang, M. Ziad, and G. Shankaranarayanan. IP-MAP: representing the manufacture of an information product. In *Proceedings of the Eight International Conference on Information Quality (ICIQ-00)*, Cambridge, MA., November 2000.

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