Estimating the Quality of Databases

Amihai Motro[1] and Igor Rakov[2]

[1] Information and Software Engineering Department
George Mason University, Fairfax, VA 22030, USA,
ami@gmu.edu, http://www.ise.gmu.edu/~ami

[2] General Electric Information Services
401 N. Washington Street MNB1D, Rockville, MD 20850, USA
igor.rakov@geis.ge.com

Abstract. With more and more electronic information sources becoming widely available, the issue of the quality of these often-competing sources has become germane. We propose a standard for specifying the quality of databases, which is based on the dual concepts of data soundness and data completeness. The relational model of data is extended by associating a quality specification with each relation instance, and by extending its algebra to calculate the quality specifications of derived relation instances. This provides a method for calculating the quality of answers to arbitrary queries from the overall quality specification of the database. We show practical methods for estimating the initial quality specifications of given databases, and we report on experiments that test the validity of our methods. Finally, we describe how quality estimations are being applied in the Multiplex multidatabase system to resolve cross-database inconsistencies.

1 Data Quality

What is it? Data quality has several dimensions. The most common dimension is *soundness*: whether the data available are the true values. Data soundness is often referred to as correctness, precision, accuracy, or validity. Another common dimension of data quality is its *completeness*: whether all the data are available. In the terminology of databases, data completeness refers to both the completeness of files (no records are missing), and to the completeness of records (all fields are known for each record). Soundness and completeness are two orthogonal dimensions, in the sense that they are concerned with completely separate aspects of data quality [6]. Other important dimensions often discussed are *consistency* and *currency*. Consistency (often referred to as integrity) assumes specific *constraints* that state the proper relationships among different data elements are known, and considers whether these constraints are satisfied by the data. Currency considers whether the data are up-to-date, reflecting the most recent values. For additional discussion of various quality dimensions, see [3,12].

T. Andreasen, H. Christiansen, and H.L. Larsen (Eds.): FQAS'98, LNAI 1495, pp. 298–307, 1998.

What can be done about it? Concerns about data quality can be addressed in three ways: Protection, measurement, and improvement. The focus of the first approach is on *prevention*. It advocates careful handling of data at every phase, from acquisition to application. In some respects, this approach is modeled after quality control methods in production processes [12]. The second approach focuses on the *estimation* of quality. A data quality measure is adopted, and the quality of the data is estimated in accordance to this measure. The third approach is the most ambitious. Data quality may be improved by identifying obvious errors in the data ("outlying values"), and either removing these values, or substituting them (as well as values that are missing altogether) with more probable values. This approach is often referred to as data *cleansing* or *scrubbing* [1].

In this paper we consider only the dimensions of data soundness and data completeness, and we focus on data quality estimation. The advantages of data quality estimation are threefold. Clearly, decision-making is improved when the quality of the data used in the decision process is known. To this end, it is important to be able to determine the quality of specific data within a given database (see Section 4). In the presence of alternative sources of information, quality estimates suggest the source that should be preferred (see Section 6). Finally, information is a commodity, and its value (price) should be based on its quality.

This paper presents an overview of the issues and the solutions proposed. For a more complete treatment the reader is referred to [8] and [9]. Our treatment of the issues is in the context of relational databases, and we assume the standard definitions of the relational model.

In Sections 2 and 3 we propose a standard for specifying the quality of relational databases, which is based on data soundness and data completeness, and we explain how such quality specifications may be estimated. Quality specifications are associated with each relation instance of the database, and may be considered an extension of the relational model. Suitably, the relational algebra is extended as well, to calculate the quality specifications of derived relation instances. This provides a method for estimating the quality of answers to arbitrary queries (Section 4). In section 5 we report on experiments that were carried out to test the validity of our methods, and in section 6 we describe how quality estimations are being applied in the Multiplex multidatabase system to resolve cross-database inconsistencies. Additional research directions are discussed briefly in Section 7.

2 Simple Quality Estimation

A database and the real world that it models can be formalized as two *instances* of the same database scheme. We denote the *actual* (stored) database instance D, and we denote the *ideal* (real-world) database instance W. Of course, W is a hypothetical instance which is unavailable. The stored instance D is an *approximation* of the ideal instance W.

How good is this approximation? Clearly, the answer to this question would provide a measure of the quality of the given database.

To determine the goodness of the approximation we must measure the *similarity* of two database instances. Since each instance is a set (of tuples), we can use measures that compare two sets of elements. One such measure is based on two components as follows (X denotes the cardinality of a set X:

- Soundness (of the database relative to the real world): $\frac{|D \cap W|}{|D|}$
- Completeness (of the database relative to the real world): $\frac{|D \cap W|}{|W|}$

Soundness measures the proportion of the stored information that is true, and completeness measures the proportion of the true information that is stored. These measures are similar to the *precision* and *recall* measures in the field of information retrieval, where they are used to rate the goodness of answers extracted from an information source in response to queries [10].

Note that each intersection element (a database tuple that is also in the real world instance) contributes to both soundness and completeness. Similarly, when a database tuple and a real world tuple differ (even in a single attribute), both soundness and completeness are affected adversely. Consider, for example, a relation Employee consisting of 20 attributes, and assume that the stored instance and the ideal instance each have 100 tuples. Assume further that the two instances are identical in all values except in the Date_of_Birth attribute, where half the values are incorrect. Soundness and completeness will both measure 0.5, whereas intuitively the quality is much higher.

A simple improvement is to *decompose* all relations to smaller units of information. A relation $(A_1, \ldots, A_n)$, in which A_1 is the key attribute, is decomposed to n relations of two attributes each: $(A_1, A_i), i = 1, \ldots, n$. Comparing decomposed instances provides measures that are more refined. In the above example, soundness and completeness will both measure 0.975.

As both D and W may be very large, the estimations of soundness and completeness should be based on *samples* of D and W.

Estimating Soundness. To determine the proportion of the stored information that is true, the following procedure is used:

1. Sample D.
2. For each $x \in D_{\text{sample}}$ determine if $x \in W$.
3. Calculate the soundness estimate as the number of "hits" divided by the sample size $|D_{\text{sample}}|$.

Step 1, sampling a database, is simple. But Step 2 is difficult: we need to determine the presence of database elements in W without actually constructing it. This is accomplished by *human verification* of the sample elements.

Estimating Completeness. To determine the proportion of the true information that is stored, the following procedure is used:

1. Sample W.
2. For each $x \in W_{\text{sample}}$ determine if $x \in D$.
3. Calculate the completeness estimate as the number of "hits" divided by the sample size $|W_{\text{sample}}|$.

Step 2, verifying the presence of elements in the database, is simple. But Step 1 is difficult: we need to sample W without actually constructing it. This may be interpreted as constructing a *fair challenge* to the database by using various resources (e.g., by judicious sampling of alternative databases).

The soundness and completeness estimates obtained by these procedures amount to a *simple quality specification* for the database.

3 Refined Quality Estimation

Information sources are rarely of uniform quality. An information source may provide information of excellent quality on one topic, and information of poor quality on another topic. Hence, the *overall* quality estimations described in the previous section may prove to be very crude estimates of the quality of *specific* data.

A substantial improvement over these methods is to *partition* the database into areas that are highly homogeneous with respect to their quality: any subarea of a highly homogeneous area would maintain roughly the same quality as the initial area. These partitions, and the corresponding quality measurements of their components, would provide a more *refined* quality specification for the database. Formal definitions for "areas" and "homogeneity" follow.

"Areas" and "subareas" are defined with *views*. Views may involve selection and projection (projections must retain the key attributes). *Homogeneity (with respect to soundness)* of a view v is defined as the average soundness difference between the given view and all its possible subviews:

$$H_S(v) = 1/N \sum_{i=1}^{N} |s(v) - s(v_i)|$$

where $v_1, \ldots, v_n$ are the possible subviews of v, and $s(v)$ is the soundness of v. A low homogeneity measure indicates that the view is highly homogeneous. It can be shown that perfect homogeneity, $H_S(v) = 0$, is achieved only when the view elements are all true or all false. A threshold value of H is used to assure that all the views of a partition are highly homogeneous. Homogeneity with respect to completeness is defined analogously.

The homogeneity measure H is easy to rationalize, but is expensive to compute. Our overall approach, however, does not depend on a specific measure, and cheaper-to-compute measures could be substituted. The alternative homogeneity measure that we use is the *Gini Index*, known from statistical classification theory [2,4]. The Gini index for the soundness of a view v is

$$2p_0 \cdot p_1 = 2p_1(1 - p_1) = 2p_0(1 - p_0)$$

where p_0 is the proportion of the incorrect elements in v, and p_1 is the proportion of the correct elements in v. The Gini index for the completeness of a view is defined analogously.

A view v would be considered *highly homogeneous* if all the possible ways of splitting v into two subviews v_1 and v_2 will not yield substantial (i.e., above some threshold) improvement in the homogeneity. The improvement in homogeneity is measured by the difference between the Gini index of the view v and the combined Gini index of the subviews v_1 and v_2:

$$\Delta G = G(v) - \alpha_1 G(v_1) - \alpha_2 G(v_2)$$

where $G(v)$ is the Gini index of v, and $\alpha_i = |v_i|/|v|$.

A recursive algorithm for finding highly homogeneous views is outlined below. Note that the algorithm must be performed separately for soundness and completeness, and that it is repeated for for each relation. Note also that it is performed only on samples. The partition views that are discovered by the algorithm are considered to projections (that retain key attributes) and selections.

Algorithm for Finding Quality Specifications

- **Input**: A sample of the relation; a threshold value.
- **Output**: A set of views on the relation that are highly homogeneous with respect to a quality measure.
- **Step 1**: Label the relation as the root node, and place it in a queue Q.
- **Step 2**: Retrieve the next node from Q, and make it the current node v.
- **Step 3**: Consider all the possible splits of v into two subnodes v_1 and v_2 and measure the reduction ΔG of each split.
- **Step 4**: Select a split that maximizes the reduction, and split v.
- **Step 5**: If the reduction of this split is greater than the threshold value, place v_1 and v_2 in Q. Otherwise, make v a leaf node and go to Step 6.
- **Step 6**: If Q is not empty, go to Step 2. Otherwise, stop.

The threshold value is needed to prevent splitting down to individual elements. Admittedly, Step 3 is computationally expensive and should be refined to reduce search space and make use of heuristic information.

The highly homogeneous partitions obtained with this algorithm and the soundness or completeness estimates of their views amount to a *refined quality specification* for the given database.

4 Estimating the Quality of Answers to Queries

An important application of database quality specifications is to infer from them quality specifications for answers issued from the database in response to arbitrary queries.

To this end we consider each query as a *chain* of individual relational operations. Each relational operation is extended to produce (in addition to the

standard result) a *quality specification of the result*. Consequently, each operation in the chain receives a quality specification from the previous operation and uses it to derive a quality specification for the next operation. This allows us to calculate quality specifications of sequences of relational operations (i.e., general queries). The relational operations that have been considered so far are Cartesian product and a combined selection-projection operation.

In other words, the traditional relational model is *extended* to include quality specifications for each relation instance, and the traditional relational algebra is extended to derive quality specifications of the results of its operations. These extensions assume that all quality specifications use views that are perfectly homogeneous; i.e., each of their subviews inherits the very same quality.

At the final step, the *overall quality* of the answer is computed from the final quality specification as follows. The views of a quality specification *partition* the answer. Hence, each answer is partitioned into views with known (inherited) quality. The overall quality is then computed as a weighted sum of the quality of these views. If we have only *estimates* of quality, then the *variance* of these estimates is also calculated. The details of these extensions to the relational model may be found in [8] and [9].

Quality specifications require partitions whose views are highly homogeneous. (A requirement for perfect homogeneity may result in views that contain very few tuples, yielding specifications that have impractically large number of views.) On the other hand, the extensions to the relational algebra assumed that the partition views are perfectly homogeneous. Nonetheless, it can be shown that these extensions are still acceptable; namely, the result of a projection-selection on a highly homogeneous view will be (with high probability) also highly homogeneous, and the homogeneity of the result of a Cartesian product will be close to the largest (worst) of the homogeneities of the input views.

5 Experimentation

The viability and performance of the approach and methods described in Sections 3 and 4 were tested in an experiment. The experiment used synthetic data and was limited to selection-projection queries (i.e., without Cartesian products). Only soundness was considered.

For this experiment a simple interactive tooolkit was constructed. The toolkit leads its users through the following steps:[1]

1. The user selects a data set, an error pattern, and a sampling rate.
2. The data set is sampled.
3. A quality (soundness) specification is derived.
4. The user submits an arbitrary query.
5. The answer is computed.
6. The quality of the answer is estimated.
7. The estimate is compared with the actual quality.

[1] The toolkit can be tried at `http://www.ise.gmu.edu/~irakov/quality/experiment.html`.

The pre-assignment of errors to the selected data set in Step 1 is the only deviation from the actual estimation process proposed in Section 3 and 4. It has two obvious advantages: it allows skipping the manual verification, and it provides the actual quality (normally unavailable) for comparison.

The measure of performance used in comparing the estimated quality to the actual quality is the *relative error*; i.e., the difference between the computed quality and the actual quality, divided by the actual quality.

Our experimentation lead to several observations. As expected, the error decreases as the sample size increases. Also, the error decreases as the answer cardinality increases. Determining the optimal threshold value for the discovery of the homogeneous views is less straightforward; our experiments indicate that it is often in the range 0.3–0.5. With sample size of about 15% and answer cardinality of about 400 tuples, the relative error is in the range of 4%–6%.

6 Application: Harmonization of Inconsistencies

In recent years there has been overwhelming interest in the problem of multidatabase integration: How to provide integrated access to a collection of autonomous and heterogeneous databases [5,11]. The usual focus in systems that address this problem is the resolution of *intensional inconsistencies*: How to reconcile the schematic differences among the participating databases (this issue is known as semantic heterogeneity). The *Multiplex* project [7] also addresses the problem of *extensional inconsistencies*: How to reconcile the data differences among the participating databases. Specifically, Multiplex does not assume that overlapping information sources are mutually consistent, and it focuses on methods for *harmonizing* inconsistent answers into a single authoritative answer.[2]

The architecture of Multiplex centers on a global database scheme that is specified in the relational model. Individual information providers may define any number of *views* over this global scheme, committing to provide the contents of these views opon request. The participating systems may have vastly different architectures (i.e., not necessarily relational), but they must deliver their contributions in the form of tables, which extend the global views that they agreed to provide. A query over the global relations must then be translated into an equivalent query over the contributed view definitions. Possibly, some queries might not have a translation, because some of the information they target has not been provided by any source; such queries would be given only *partial answers*. On the other hand, it is possible that some queries would have *multiple answers*, because some of the information they target is provided by more than one source; such answers would require *harmonization*.

Essentially, the Multiplex approach is to allow a multidatabase designer to specify a-priori a *conflict resolution strategy* whenever the potential for inconsistencies exists (i.e., whenever the definitions of the contributed views overlap). One important option in this strategy (among many other options) is to base the resolution on the quality of the alternatives.

[2] The Multiplex server is available at `http://www.ise.gmu.edu/~multiplex`.

Assume a query Q has multiple answers $A_1, \ldots A_n$. Assume first that there is no information regarding their quality; i.e., each answer is presented by its source as if it is sound and complete. This may be formalized as each answer being presented as both sound and complete: $s(A_i) = 1.0$ and $c(A_i) = 1.0$ for $i = 1, \ldots, n$. Obviously, unless these answers are identical, these soundness and completeness claims are mutually inconsistent.

An intuitive heuristic is to let the answer providers *vote* on the complete answer space $\bigcup_{i=1}^{n} A_i$ as follows:

1. If $x \in A_i$ then provider i votes "yes" on x (an expression of the claimed soundness of A_i).
2. If $x \notin A_i$ then provider i votes "no" on x (an expression of the claimed completeness of A_i).

Where x is an element of the answer space. Define:

1. $L = \{x | x$ received only $''$yes$''$ votes$\}$
2. $U = \{x | x$ received at least one $''$yes$''$ vote$\}$

The true answer A is "sandwiched" between L and U: $L \subseteq A \subseteq U$. Hence, L and U constitute a *lower bound* and an *upper bound* for the true answer A. This "interval" conforms nicely with how unknown values are estimated in other disciplines. If desired, it is possible to create in-between layers E_i, as well

$$E_i = \{x | x \text{ received at least } i \ ''\text{yes}'' \text{ votes}\}$$

thus providing "tiers" within the largest estimate U.

Assume now that each answer A_i has its individual quality specifications: $s(A_i) = s_i$ and $c(A_i) = c_i$. Using probabilistic arguments, the voting scheme is extended:

1. If $x \in A_i$ then provider i votes on x

$$P(x \in A | x \in A_i) = s_i$$

2. If $x \notin A_i$ then provider i votes on x

$$P(x \in A | x \notin A_i) = \frac{(s_i/c_i) - s_i}{(n/|A_i|) - 1}$$

where n is the size of the entire answer space.

Votes are then combined to provide *ranking* of the tuples in the answer space. This is useful for handling queries that pre-specify the desired answer cardinality.

7 Research Directions

We introduced a new model for data quality in relational databases, which is based on the dual measures of soundness and completeness. This model extends the relational model by assigning each relation instance a quality specification, and by extending the relational algebra to calculate the quality specifications of derived relation instances. The principal purpose of this model is to provide answers to arbitrary queries with estimations of their quality. We described an algorithm for deriving initial quality specifications for the stored relations from samples whose quality had been established manually. And we described how quality estimations are being applied in the Multiplex multidatabase system to resolve cross-database inconsistencies.

There are many issues that require further investigation, and we mention here three research directions.

We discussed the advantage of considering the correctness of individual attributes over the correctness of entire tuples. Still, an individual value is either correct or incorrect, and, when incorrect, we do not consider the amount by which it deviates from the true value. A possible extension of this work is to generalize our quality specifications and extend our quality estimations so that they consider the *similarity* of the stored values to the real values.

Because of the cost of establishing quality estimations, our methods are most suitable for static information. When the information is dynamic, it would be advisable to timestamp the estimations at the time that they were obtained and attach these timestamps to all quality inferences. One may also consider the automatic attenuation of quality estimations as time progresses.

The cost of establishing quality estimations is due mainly to the human effort that it requires. An attractive alternative is to develop methods that "discover" the quality "automatically" (or, more accurately, with only limited human guidance). A possible direction is to draw on techniques that have been developed for data mining and knowledge discovery.

Acknowledgment. This work was supported in part by DARPA grant N0060-96-D-3202.

References

1. Bort, J.: Scrubbing dirty data. *InfoWorld*, 17(51), December 1995.
2. Breiman, L., Friedman, J., Olshen, R., and Stone, Ch.: *Classification and Regression Trees*. Wadsworth International Group, 1984.
3. Fox, C., Levitin, A., and Redman, T.: The notion of data and its quality dimensions. *Information processing and management*, 30(1), 1994.
4. Chen, M. C., McNamee, L., and Matloff, N.: Selectivity estimation using homogeneity measurement. *Proceeding of the International Conference on Data Engineering*, 1990.
5. Hurson, A.R., Bright, M.W., Pakzad, S.: *Multidatabases: An Advanced Solution to Global Information Sharing*, IEEE Computer Society Press, 1993.

6. Motro, A.: Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480–502, December 1989.
7. Motro, A: Multiplex: A Formal Model for Multidatabases and Its Implementation. Technical Report ISSE-TR-95-103, Department of Information and Software Engineering, George Mason University, March 1995.
8. Motro, A., Rakov, I: Not all answers are equally good: Estimating the quality of database answers. In *Flexible Answering Systems* (T. Andreasen, H. Christiansen, and H.L. Larsen, Editors), Kluwer Academic Publishers, 1997, 1–21.
9. Rakov, I: *Data quality and Its Use for Reconciling Inconsistencies in Multidatabase Environments*, Ph.D. Dissertation, George Mason University, May 1998.
10. G. Salton and M. J. McGill: *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, New York, 1983.
11. Wiederhold, G. (Ed.): Special Issue of the Journal of Intelligent Information Systems, 6(2-3), June 1996.
12. Wang, R., Storey, V., and Firth, Ch.: A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4), August 1995.