Monica Scannapieco, Paolo Missier, Carlo Batini

Data Quality at a Glance

The paper provides an overview of data quality, in terms of its multidimensional nature. A set of data quality dimensions is defined, including accuracy, completeness, time-related dimensions and consistency. Several practical examples on how such dimensions can be measured and used are also described. The definitions for data quality dimensions are placed in the context of other research proposals for sets of data quality dimensions, showing similarities and differences. Indeed, while the described core set of dimensions is shared by most proposals, there is not yet a common standard defining which are the data quality component dimensions and what is exactly their meaning.

1 Introduction

The consequences of poor quality of data are often experienced in everyday life, but without making the necessary connections to its causes. For example, the late or missed delivery of a letter is often blamed on a dysfunctional postal service, although a closer look often reveals data-related causes, typically an error in the address, originating in the address database. Similarly, the duplicate delivery of automatically generated post is often indicative of a database record duplication error. Inaccurate and duplicate addresses are examples of *data quality* problems.

Awareness of the importance of improving the quality of data is increasing in many contexts. In the public sector, for instance, a number of e-Government initiatives address data quality issues both at European and national levels. The European directive 2003/98/CE on the reuse of public data [EU Directive 2003] highlights the importance of reusing the vast data assets owned by public agencies. A first necessary step for data reuse is to guarantee its quality through *data cleaning* campaigns, in order to make it attractive to potential new users and customers.

A second important reason for addressing data quality problems is the growing need to integrate information across disparate data sources, because poor quality hampers integration efforts. This is a well-known problem in data warehousing, in that much of the implementation budget is spent on data cleaning activities. In a data warehouse, integrated data are materialized, as opposed to *virtual* data integration, where data are presented to the user by a unique virtual view, though being physically stored in disparate sources. Virtual data integration is a recent phenomenon that has been growing alongside Web communities and it is also affected by data quality problems, because inconsistencies in data stored at different sites make it difficult to provide integrated information as results of user queries. Indeed, when collecting data from sources to answer a user query, if inconsistencies occur, data must be reconciled *on the fly* in order to provide a suitable result to the query.

From the research perspective, data quality has been addressed in different contexts, including statistics, management and computer science. Statisticians were the first to investigate some of the problems related to data quality, by proposing a mathematical theory for considering duplicates in statistical data sets, in the late 60's [Fellegi & Sunter 1969]. They were followed by researchers in management, who, at the beginning of the 80's, focused on how to control data manufacturing systems[1] in order to detect and eliminate data quality problems (see as an example [Ballou & Pazer 1985]). Only at the beginning of the 90's computer scientists began considering the problem of defining, measuring and improving the quality of electronic data, stored in databases, data warehouses and legacy systems.

1. Like traditional product manufacturing systems, data manufacturing systems manage the life cycle of data as *information products*.

Data quality can be intuitively characterized as *fitness for use* [Wang 1998]. However, in order to fully understand the concept, researchers have traditionally identified a number of specific quality *dimensions*. A dimension or characteristic captures a specific facet of quality. The more commonly referenced dimensions include accuracy, completeness, currency and consistency, although many other dimensions have been proposed in the literature, as described in the next sections.

In this paper, we first introduce the notion of data quality, highlighting its multidimensional nature (Section 2). Then, we show several examples on data quality and on how quality problems can be detected (Section 3); later, we describe how the provided definitions for data quality dimensions are placed in the context of research proposals for sets of data quality dimensions (Section 4). The contribution of the paper is finally summarized in Section 5.

2 A Set of Data Quality Dimensions

When people think about data quality, they usually only refer to accuracy. Indeed, data are normally considered of poor quality if typos are present or wrong values are associated to a concept instance, such as a person's erroneous birth date or age. However, data quality is more than simply data accuracy. Other significant dimensions such as completeness, consistency and currency are necessary in order to more fully characterize the quality of data.

We are going to illustrate the meaning of such dimensions with reference to the example relation shown in figure 1.

ID	Title	Director	Year	#Remakes	LastRemakeYear
1	Casablanca	Weir	1942	3	1940
2	Dead Poets Society	Curtiz	1989	0	NULL
3	Rman Holiday	Wylder	1953	0	NULL
4	Sabrina	NULL	1964	0	1985

Fig. 1: A relation *Movies* with data quality problems

In the figure, a relation *Movies* is shown and the cells with data quality problems are shadowed. At a first glance, only the cell corresponding to the title of movie 3 seems to be affected by an accuracy problem, i.e., a misspelling in the title (*Rman* instead of *Roman*). However, we note that a swap in the directors of movies 1 and 2 also occurred (Curtiz directed movie 1, and Weir movie 2), which is also considered an accuracy issue. The other shadowed cells show further quality problems: a missing value for the director of movie 4 (completeness), and a *0* value for the number of remakes of the same movie. Specifically, for the movie 4 a remake was actually made in 1985, therefore the *0* value for the *#Remakes* attribute may be considered a currency problem, i.e., the remake has not yet been recorded in the database. Finally, there are two consistency problems: first, for movie 1, the value of *LastRemakeYear* cannot be lower than *Year*; second, for movie 4, the value of *LastRemakeYear* and *#Remakes* are inconsistent, i.e., either *#Remakes* is not *0* or *LastRemakeYear* is *Null.*

The example shows that:

- data quality is a multi-faceted concept, to the definition of which different dimensions concur;
- accuracy can be easily detected in some cases (e.g. misspellings) but it is more difficult to be detected in other cases (e.g. whereas the values are admissible but not correct);
- a simple example of completeness error has been shown, but similarly to accuracy, also completeness can be very difficult to evaluate (imagine for instance that a whole movie tuple is missing from the relation);
- the currency dimension measures the fact that a value is out of date;
- detecting an inconsistency may not be sufficient to determine which record is at fault, i.e. for movie 1, which of Year or LastRemakeYear is wrong?

In the following sections we will define accuracy, completeness, currency and consistency more precisely. These four dimensions are only some of a large set of dimensions proposed in the data quality literature, as discussed in Section 4; for instance, many subjective dimensions have also been proposed to characterize data quality, including, among others, reputation, objectivity, believability, interpretability.

2.1 Accuracy

Accuracy can be evaluated for disparate granularity levels of a data model, ranging from single values to entire databases. For single data values, accuracy measures the distance between a value *v* and a value *v'* which is considered correct. Two kinds of accuracy can be identified, namely a *syntactic accuracy* and a *semantic accuracy.*

Syntactic accuracy is measured by means of *comparison functions* that evaluate the distance between *v* and *v'*. Edit distance is a simple example of comparison function, taking into account the cost of converting a string *s* to a string *s'* through a sequence of character insertions, deletions, and replacements. More complex comparison functions exist which take into account similar sound, transpositions etc. (see, e.g., [Elfekey et al. 2002] for a short review).

As an example, let us consider again the relation *Movies*, shown in figure 1. The accuracy error of movie 3 on the *Title* value is a syntactic accuracy problem. As the correct value for *Rman Holidays* is *Roman Holidays*, the edit distance between the two values is equal to 1 and simply corresponds to the insertion of the char »o« in the string *Rman Holidays*.

Semantic accuracy captures the cases in which *v* is a syntactically correct value, but it is different from *v'*.

In the same *Movies* relation, swapping the directors' names for tuples 1 and 2 results in a semantic accuracy error, because although a director named *Weir* would be syntactically correct, he is not the director of *Casablanca*, therefore the association between movie and director is semantically inaccurate.

From these examples, it is clear that detecting semantic accuracy is typically more involved than detecting syntactic accuracy. It is often possible to detect semantic inaccuracy in a record, and to provide an accurate value, by comparing the record with equivalent data in different sources. This requires the ability to recognize that two records refer to the same *real world entity*, a task often referred to as the *object identification problem* (also called record matching or record linkage) [Wang & Madnick 1989]. As an example, if two records store *J.E. Miller* and *John Edward Miller* as a person's name, the object identification problem aims to realize if the two records represent the same person or not. There are two aspects to this problem:

- Identification: records in different sources have typically different identifiers. Either it is possible to map identification codes (when available), or *matching keys* must be introduced to link the same records in different sources.
- Decision: once records are linked on the basis of a matching key, a decision must be made as to whether or not the records represent the same real world entity.

The version of accuracy discussed above, both syntactic and semantic, refers to a single value, for instance of a relation attribute. In practical cases, coarser accuracy metrics may be applied. As an example, it is possible to compute the accuracy of an attribute (column accuracy), or of a relation or of a whole database.

When considering a coarser granularity than values, there is a further notion of accuracy that needs to be introduced for some specific data model, namely *duplication*. Duplication occurs when a real world entity is stored twice or more in a data source. Of course, when a primary key consistency check is performed when populating a source, the duplication problem does not occur. However, for files or other data structures that do not allow defining such type of constraints the duplication problem is very important and critical. A typical cost due to duplication is the mailing cost that enterprises must pay for mailing to their customers, whenever customers are stored more than once in the enterprise's database. To this direct cost, also an indirect cost must be added that consists of the image loss to the enterprise with respect to its customers that are bothered by multiple mailings.

For relations and database accuracy, a *ratio* is typically calculated between accurate values and total number of values. So, for instance, the accuracy of a relation can be measured as the ratio between the number of accurate cell values and the total number of cells in the table.

More complex metrics can be defined. For instance, as said, a possibility for accuracy evaluation is to match tuples from the source under examination with tuples of another source which is assumed to contain the same but correct tuples. In such a process, accuracy errors on attribute values can be either such that they do not affect the tuple matching, or they can prevent the process itself, not allowing the matching. Therefore, metrics that consider the »importance« of accuracy errors on attribute values with respect to the impact in the matching process need to be introduced. As an example, given a *Person* record, an accuracy error on a *Fiscal Code* value can be considered more important than an accuracy error on the *Residence Address*, as it can prevent the record from being matched.

2.2 Completeness

Completeness can be generically defined as »the extent to which data are of sufficient breadth, depth and scope for the task at hand« [Wang & Madnick 1989].

In [Pipino et al. 2002], three types of completeness are identified. *Schema completeness* is defined as the degree to which entities and attributes are not missing from the schema. *Column completeness* is a function of the missing values in a column of a table. *Population completeness* amounts to evaluating missing values with respect to a reference population.

If focusing on a specific data model, a more precise characterization of completeness can be given. Specifically, in the relational model completeness can be characterized with respect to the presence and meaning of null values.

In a model *without* null values, we need to introduce the concept of *reference relation*. Given the relation *r*, the reference relation of *r*, called *ref(r)*, is the relation containing all tuples that satisfy the relational schema of *r*.

As an example, if *dept* is a relation representing the employees of a given department, and a given employee of the department is not represented as a tuple of *r*, then the tuple corresponding to the missing employee is in *ref(r)*. In practical cases, the reference relations are rarely available, instead their cardinality is much easier to get. There are also cases in which the reference relation is available but only periodically (e.g. when a census is performed).

On the basis of the reference relation, in a model without null values, completeness is defined as the fraction of tuples actually represented in a relation *r*, with respect to the whole number of tuples in *ref(r)*, i.e. :

```
Cardinality of r/
Cardinality of ref(r)
```

In a model *with* null values, the presence of a null value has the general meaning of a missing value. In order to characterize completeness, it is important to understand *why* the value is missing. Indeed, a value can be missing either because it exists but is unknown, or because it does not exist at all, or because its existence is unknown.

Let us consider, as an example, a *Person* relation, with the attributes *Name*, *Surname*, *BirthDate*, and *Email*. The relation is shown in figure 2.

For tuples with ID 2, 3 and 4, the email value is *null*. Let us suppose that the person represented by tuple 2 has no email; in this case, there is no incompleteness. If the person represented by tuple 3 has an email but it is not known which is the value, than tuple 3 is incomplete. Finally, if it is not known whether the person represented by tuple 4 has an email or not, we cannot determine whether the tuple is incomplete.

Besides null values meaning, precise definitions for completeness can be provided by considering the granularity of the model elements, i.e., value, tuple, attribute and relations, as shown in figure 3.

Specifically, it is possible to define:

- a *value completeness* to capture the presence of null values for some attributes of tuples;
- a *tuple completeness* to characterize the completeness of a whole tuple with respect to the values of all attributes;
- an *attribute completeness* to measure the number of null values of a specific attribute in a relation;
- a *relation completeness* that captures the presence of null values in the whole relation.

As an example, let us consider figure 4, in which a *Students* relation is shown. The tuple completeness evaluates the percentage of specified values in the tuple with respect to the total number of attributes of the tuple itself. Therefore, in the example, the tuple completeness is: 1 for tuples 6754 and 8907, 4/5 for tuple 6578, 3/5 for tuple 0987 etc.

One way to see the tuple completeness is as a measure of the information content carried on by the tuple with respect to the maximum potential information content of the tuple. With reference to this interpretation, we are implicitly assuming that all values of the tuple equally contribute to the total information content of the tuple. Of course, this may be not the case, as different applications can weight differently the attributes of a tuple.

The attribute completeness evaluates the percentage of specified values in the column corresponding to the attribute with respect to the total number of values that should have been specified. In figure 4, let us consider an application that computes the average of votes obtained by students. The absence of some values for the *Vote* attribute affects the result but does not preclude the computation itself, therefore a characterization of the *Vote* completeness may be useful, as it allows associating a certain confidence to the average of votes computation. Relation completeness is relevant in all the applications that need to evaluate the completeness of a whole relation and can tolerate the presence of null values on some attributes. It measures how much information is represented by the relation, by evaluating the actually available informa-

ID	Name	Surname	BirthDate	Email
1	John	Smith	03/17/1974	smith@abc.it
2	Edward	Monroe	02/03/1967	NULL
3	Anthony	White	01/01/1936	NULL
4	Marianne	Collins	11/20/1955	NULL

Not Existing

Existing But Unknown

Not Known If Existing

Fig. 2: Examples of different NULL value meanings

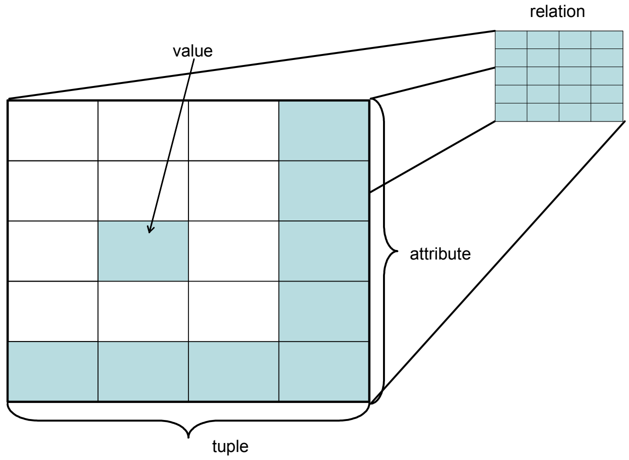

Fig. 3: Completeness of different elements of the relational model

StudentID	Name	Surname	Vote	ExaminationDate
6754	Mike	Collins	29	07/17/2004
8907	Anne	Herbert	18	07/17/2004
6578	Julianne	Merrals	NULL	07/17/2004
0987	Robert	Archer	NULL	NULL
1243	Mark	Taylor	26	09/30/2004
2134	Bridget	Abbott	30	09/30/2004
6784	John	Miller	30	NULL
0098	Carl	Adams	25	09/30/2004
1111	John	Smith	28	09/30/2004
2564	Edward	Monroe	NULL	NULL
8976	Anthony	White	21	NULL
8973	Marianne	Collins	30	10/15/2004

Fig. 4: Example of completeness of tuples, attributes, relations

tion content with respect to the possible one, i.e. without null values. According to this interpretation, completeness of the relation Student in figure 4 is 53/60. Let us notice that a further important notion of relation completeness is possible if we admit an open world assumption, i.e., it is not true that the stored tuples are *all* the tuples satisfying the schema. In the example, let us suppose that 3 students actually passed the examination but they were not recorded in the relations. In this case, the completeness of the relation *Students* is 12/15.

2.3 Time-related Dimensions: Currency, Timeliness and Volatility

An important aspect of data is how often they vary in time. There are *stable* data that typically do not need any update; examples are attributes such as birth date, surnames, eye color. On the contrary, there are many examples of *time-variable* data, such as ages, addresses, salaries, and so on. In order to capture aspects concerning temporal variability of data, different data quality dimensions need to be introduced. The principal time-related dimensions are currency, timeliness and volatility.

Currency measures how promptly data are updated. In the example shown in figure 1, the attribute *#Remakes* of the movie 4 is not current because a remake of the movie 4 had been performed, but this information did not result in an increased value for the number of remakes. Similarly, if a residence address of a person is updated, i.e. it actually corresponds to the address where the person lives, then it is current.

Volatility measures the frequency according to which data vary in time. For instance, stable data such as birth dates have the lowest value in a given metric scale for volatility, as they do not vary at all. Conversely, stock quotes have high volatility values, due to the fact that they remain valid for very short time intervals.

Timeliness measures how current data are, relative to a specific task. It is motivated by the fact data it is possible to have current data that are actually useless because they are *late* for a specific usage. For instance, if considering a timetable for university courses: it can be current, thus containing the most recent data, but it can be not timely, if it only becomes available after the start of lessons.

Volatility is a dimension that inherently characterizes types of data. Therefore, there is no need of introducing specific metrics for it.

Currency is typically measured with respect to *last update* metadata, i.e., the last time in which the specific data have been updated. For data types that change with a fixed frequency, last update metadata allow to compute currency straightforwardly. For data types whose change frequency can vary, one possibility is to calculate an average change frequency and perform the currency computation with respect to it, admitting error rates. As an example, if a data source stores residence addresses that are estimated to change each five years, then an address with a last update metadata reporting a date corresponding to one month before the observation time, can be estimated to be *current*. Instead, if the last update metadata reports a date corresponding to ten years before the observation time, it can be estimated as possibly *not current*. Notice that, accuracy and currency are very similarly perceived, namely non current data are often perceived as non accurate ones. Nevertheless, it is important to distinguish the two dimensions, which are inherently very different and therefore require specific improvement solutions.

Timeliness measurement implies that not only data are current, but are also in time for a specific usage. Therefore, a possible measurement consists of *(i)* a currency measurement and *(ii)* a check if data are available *before* the planned usage time.

More complex metrics can be defined for computation of time-related dimensions. We cite, as an example, the metric defined in [Ballou & Pazer 2003], in which the three dimensions timeliness, currency and volatility are linked together by defining timeliness as a function of currency and volatility. More specifically, currency is defined as:

```
Currency = Age +
(DeliveryTime - InputTime)
```

where *Age* measures how old the data unit is when received, *DeliveryTime* is when the information product is delivered to the customer and *InputTime* is when the data unit is obtained. Therefore, currency is the sum of how old are data when received (*Age*) plus a second term that measures how long data have been in the information system. Volatility is defined as the length of time data remains valid. Timeliness is defined as:

```
max{0, 1- currency/
    volatility}
```

Timeliness ranges from 0 to 1, where 0 means bad timeliness and 1 means a good timeliness. The importance of currency depends on volatility: data that are highly volatile must be current while currency is not important for data with a low volatility.

2.4 Consistency

The consistency dimension captures the violation of semantic rules defined over (a set of) data items. With reference to the relational theory, *integrity constraints* are an instantiation of such semantic rules.

Integrity constraints are properties that must be satisfied by all instances of a database schema. It is possible to distinguish two main categories of integrity constraints, namely: intra-relation constraints and inter-relation constraints.

Intra-relation integrity constraints can regard single attributes (also called domain constraints) or multiple attribute of a relation. As an example of intra-relation integrity constraint, let us consider the *Movies* relation of the example shown in Figure 1; as already remarked, the *Year* attribute values must be lower than the *LastRemakeYear* attribute values.

As an example of inter-relations integrity constraint, let us consider the *Movies* relation again, and the relation *OscarAwards*, specifying the oscar awards won by each movie, and including an attribute *Year* corresponding to the year when the award was assigned. An example of inter-relation constraint states that *Movies.Year* must be equal to *OscarAwards.Year.*

Integrity constraints have been largely studied in the database research area, and the enforcement of dependencies (e.g. key dependency, functional dependency, etc.) is present in modern database systems. The violation of integrity constraints in legacy database systems can be quite easily checked from an application layer encoding the consistency rules. Also, most of the available cleaning tools, allow the definition of consistency rules that can automatically be checked.

So far, we have discussed integrity constraints in the relational model as an instantiation of consistency semantic rules. However, consistency rules can still be defined for non-relational data. As an example, in the statistical area, some data coming from census questionnaires have a structure corresponding to the questionnaire *schema*. The semantic rules are thus defined over such a structure, in a way which is very similar to relational constraints definition. Of course, such rules, called *edits*, are less powerful than integrity constraints because they do not rely on a data model like the relational one. Nevertheless, data editing has been done extensively in the national statistical agencies since the 1950s and is defined as the task of detecting inconsistencies by formulating rules that must be respected by every correct set of answers. Such rules are expressed as edits that encode error conditions.

As an example, an inconsistent answer can be to declare *marital status as married and age as 5 years old*. The rule to detect this kind of errors could be the following: *if marital status is married, age must be not less than 14*. The rule must be put in form of an edit, which expresses the error condition, namely:

```
(marital status = married)
 ∧ (age < 14)
```

After detection of erroneous records, the act of correcting erroneous fields by restoring correct values is called *imputation*. The problem of localizing errors by means of edits and imputing erroneous field is usually referred to as the edit-imputation problem.

The Fellegi-Holt method [Fellegi & Holt 1976] is a well-known theoretical model for editing with the following three main goals, namely:

- The data in each record should satisfy all edits by changing the fewest fields.
- Imputation rules should be derived automatically from edits.
- When imputation is necessary it is desiderable to maintain the marginal and joint frequency distribution of variables.

The interested reader can find a review of methods for practically solving the edit-imputation problem in [Winkler 2004].

2.5 Tradeoffs among Dimensions

Data quality dimensions are not independent of each other but correlations exist among them. If one dimension is considered more important than the others for a specific application, than the choice of favoring it may imply negative consequences on the others. Establishing tradeoffs among dimensions is an interesting problem, as shown by the following examples.

First, tradeoffs may need to be made between timeliness and a dimension among accuracy, completeness, and consistency. Indeed, having accurate (or complete or consistent) data may require time and thus timeliness is negatively affected. Conversely, having timely data may cause lower accuracy (or completeness or consistency). An example in which timeliness can be preferred to accurate, complete or consistent data is given by most Web applications. As the time constraints are often very stringent for web available data, it may happen that such data are deficient with respect to other quality dimensions. For instance a list of courses published on a university Web site, must be timely, although there could be accuracy or consistency errors and some fields specifying courses could be missing. Conversely, if considering an e-banking application, accuracy, consistency and completeness requirements are more stringent than timeliness, and therefore delays are mostly admitted in favor of correctness of dimensions different from timeliness.

A further significant case of tradeoff is between consistency and completeness

[Ballou & Pazer 2003]. The question is: »Is it better to have few but consistent data, i.e. poor completeness, or it is better to have much more data but inconsistent i.e. poor consistency?«. This choice is again very domain specific. As an example, statistical data analysis typically requires to have significant amount of data in order to perform analysis and the approach is to favor completeness, tolerating inconsistencies, or adopting techniques to solve them. Conversely, if considering an application that calculates the salaries of a company's employees, it is more important to have a list of consistency checked salaries than a complete list, that can possibly include inconsistent salaries.

3 Examples on Detecting Quality Problems

We now present a complete example in the domain of a movies database, to further illustrate the various types of data quality dimensions previously introduced, the techniques available to detect data errors relative to those dimensions, and the corresponding metrics that can be used to describe those errors. As the example shows, the detection technique usually also suggests a method for error correction.

To each dimension, we can associate one or more metrics that express the level of quality of a data item with respect to the dimension. By extension, the quality level of a data set is expressed using some function of the individual metrics, as illustrated in the following examples.

We use Federico Fellini's filmography, as reported (correctly!) by the Internet Movie Database (URL: *http://www.imdb.com/*), to illustrate the use of each of the quality properties listed above.

Syntactic Accuracy

As we assume that the syntax check is based on a grammar, detection requires a parser, and possibly a lexical analyzer. Suppose that a simple syntax is defined for a movie title, as follows:

```
<original title> <year>
["aka" <translation>
(<Country>)]*
```

where *Country* is a literal (the exact definition of these tokens is omitted).

As an example:

```
E la nave va (1983)
aka And the Ship Sails On (USA)
aka Et vogue le navire (France)
```

Error detection is performed by parsing the titles, and additionally, using vocabularies for different languages, the language of each title may be detected, with some approximation. This step will require the computation of comparison functions, as highlighted in Section 2.1. Note that the parser has some limited capability to detect other types of errors, for instance a missing year (see completeness dimension).

Also, note that the precision of the testing procedure itself is critical to the measurement. In this case, a simple grammar may fail to parse acceptable format variations that occur in the data set, like the following:

```
Ultima sequenza, L' (2003)
aka The Lost Ending
(International: English
title)
aka Federico Fellini: I'm a
Big Liar (USA: literal
English title)
```

where the country may itself be structured.

The syntactic accuracy test can produce an error code representing a specific type of syntax error, depending on the grammar. Its extension to the data set is the frequency of each status code where the translation language may be compound.

Semantic Accuracy

Let us consider for instance that given the entire filmography for Fellini, we would like to test that a film in the set is cited correctly, by validating its fields (i.e., the date, actors, and other details including the director) against a matching record found in some reference film encyclopedia of choice. Note that test amounts to (i) finding a matching entry in the reference dataset, and (ii) validating the record fields against this entry.

To perform this test, the problem is to compare two filmographies for Fellini from two databases, trying to match the films in each. For each matching pair, test that the values conform to each other.

A suitable metric is defined by the specific syntactic accuracy criteria adopted, which may detect various types of discrepancies.

Completeness

A completeness test would determine whether or not the filmography for Fellini is complete, or whether for a single film, the actor list is complete.

This test may use the underlying record matching procedures just described, as by definition it requires a reference data source. This type of completeness checks are usually referred to as *horizontal completeness* (or relational completeness in the relational model), as opposed to *vertical completeness* (or attribute completeness in the relational model) that instead can consider how many values are available for fields that are part of the movies schema, e.g., biography items for the director, and the additional fields available for a film: runtime, country, language, certification for various countries, etc.

A vertical completeness test should be aware of the schema, and of the reasons why some values are missing. For instance, it would be a mistake to count a missing year of death as error, if the director is still alive, as discussed in 2.2, with respect to null values semantics.

A natural metric for this property counts the frequency of missing values for each field.

Currency and Timeliness

Since by definition the computed currency status is valid only at the time the test is performed, currency is typically estimated based on recent accuracy tests, and on the prediction of the next state change in the real-world entity.

To illustrate, let us consider for instance the number of awards won by a certain film. Following the actual awards ceremony, this information may not appear in the database for a certain amount of time, during which the database is not current. While a test performed during this time would reveal this error, it is also easy to determine indirectly when the awards information is current, because the award dates are known in advance: if it is updated today, the awards information remains current until the next scheduled ceremony.

Thus, a test for currency in this case would consider the last update date of the information in the database, and the scheduled data of the corresponding real-world state change.

In general, however, this latter information is not available (e.g. in the case of a director's death), hence we must rely on indirect indicators, for instance the average update frequency computed from similar information (other films, other directors), and the average lag time between these events and the database update.

In general, therefore, currency can often only be estimated, and the corresponding metric should include a confidence level in the estimate.

With respect to timeliness, we remind that it involves a user-defined deadline for restoring currency.

For example, a user may need the awards information to compile some official statistics, which are to be ready by a certain date. This determines a timeliness requirement that affects the updates procedures.

A suitable metric is the time lag between the set deadline and the time the data actually becomes current.

Consistency

Consider the following consistency rule:

```
The movie production year
must be compatible with the
director's lifetime.
```

The following movies, which appear in the filmography, do not comply with the rule (Fellini died in 1993):

```
Ultima sequenza, L' (2003)
aka The Lost Ending
(International: English title)

Fellini: Je suis un grand
menteur (2002)
aka Federico Fellini: I'm a
Big Liar (USA: literal
English title)
aka Federico Fellini: Sono un gran
 bugiardo (Italy)
```

Here we have again an example of poor detection test (as opposed to a poor quality database). Indeed, upon closer inspection, it becomes clear that Fellini did not direct these movies, but they are instead documentaries about the great director. This distinction can be made using the available movie type field (with value »Himself – filmography« in the IMD page).

A better rule for consistency testing would thus take this additional field into account. In general, a consistency test may:

1. define a rule along with its scope, i.e. to which titles it should apply;
2. apply the rule to each record in the scope.

The definition of the quality metrics may vary depending on the rule, in the example it may simply be a boolean value.

By extension, the corresponding data set metric is a count of the records that violate the rule.

4 Problems and Challenges to Data Quality Definition

In the previous sections we have introduced data quality dimensions and we have shown several examples on how they can be measured. This core set, namely accuracy, completeness, time-related dimensions and consistency, is shared by most proposals for data quality dimensions in the research literature. Such set is more suitable for some contexts rather than others. It can be adopted in e-Business and e-Government contexts, and in other contexts whereas a *general* characterization of quality of data is needed. However, for specific categories of data and for specific application domains, it may be appropriate to have more specific sets of dimensions. As an example, for geographical information systems specific, *standard* sets of data quality dimensions are under investigation (e.g. [ISO 2005]). With respect to a general set of data quality dimensions, a standard does not yet exist, but the research community has proposed various ones. In figure 5, five proposals for sets of data quality dimensions are shown: Wand 1996 [Wand & Wang 1996], Wang 1996 [Wang & Strong 1996], Redman 1996 [Redman 1996], Jarke 1999 [Jarke et al. 1999] and Bovee 2001 [Bovee et al. 2001].

Let us notice that the set of dimensions described in the previous section is common to all the proposals, but further dimensions are present in the majority of the proposals, such as interpretability, relevance/relevancy and accessibility.

A further point on which the research community is still debating is the exact meaning of each dimension. In the following, we show some contradictions and analogies by comparing some definitions

	WandWang 1996	WangStrong 1996	Redman 1996	Jarke 1999	Bovee 2001
Accuracy	X	X	X	X	X
Completeness	X	X	X	X	X
Consistency / Representational Consistency	X	X	X	X	X
Time-related Dimensions	X	X	X	X	X
Interpretability		X	X	X	X
Ease of Understanding / Understandability		X			
Reliability	X			X	
Credibility				X	X
Believability		X			
Reputation		X			
Objectivity		X			
Relevancy / Relevance		X	X		X
Accessibility		X		X	X
Security / Access Security		X		X	
Value-added		X			
Concise representation		X			
Appropriate amount of data/ amount of data		X	X		
Availability				X	
Portability			X	X	
Responsiveness / Response Time				X	

Fig. 5: Dimensions in different proposals

Wand 1996	Timeliness refers only to the delay between a change of a real world state and the resulting modification of the information system state
Wang 1996	Timeliness is the extent to which age of the data is appropriate for the task at hand
Redman 1996	Currency is the degree to which a datum is up-to-date. A datum value is up-to-date if it is correct in spite of possible discrepancies caused by time-related changes to the correct value
Jarke 1999	Currency describes when the information was entered in the sources and/or the data warehouse. Volatility describes the time period for which information is valid in the real world
Bovee 2001	Timeliness has two components: age and volatility. Age or currency is a measure of how old the information is, based on how long ago it was recorded. Volatility is a measure of information instability-the frequency of change of the value for an entity attribute

Fig. 6: Definitions of time-related dimensions

for the time-related and completeness dimensions. So, if on one hand the previously described dimensions can be considered a quite well established set, on the other hand the purpose of the following discussion is to show that the research community is still studying which is the *best* way to define data quality.

In figure 6, definitions for currency, volatility and timeliness are illustrated:

- Wand 1996 and Redman 1996 provide very similar definitions but for different dimensions, i.e. for timeliness and currency respectively. Notice that the definition for currency proposed by Redman 1996 is similar to the one proposed in Section 2.3.
- Bovee 2001 only provides a definition for timeliness in terms of currency and volatility, and Bovee 2001 currency is timeliness as defined by Wang 1996;
- volatility is defined similarly in Bovee 2001 and Jarke 1999.

The comparison shows that there is no substantial agreement on the name to use for a time-related dimension; indeed, currency and timeliness are often used to refer to the same concept. There is not even an agreement on the semantics of a specific dimension; indeed, for timeliness, different meanings are provided by different authors.

In figure 7, different proposals for completeness definition are shown.

By comparing such definitions, it emerges that :

- completeness is evaluated at different granularity levels and by different perspectives, like in Wang 1996;
- completeness is explicitly or implicitly related to the notion of quotient and collection, by measuring which fraction of a possible total is present, like in Jarke 1999.

However, there is a substantial agreement on what completeness is, though it is often tied to different granularity levels (source, attribute etc.) and sometimes to data model elements.

5 Conclusions

Quality of data is a complex concept, the definition of which is not straightforward. In the paper, we have illustrated a basic definition for it, that relies on the proposals presented in the research literature. Some metrics and examples on how to apply such metrics are also shown, with the purpose of illustrating typical steps performed to measure the quality of data. The proposed definitions for accuracy, completeness, consistency and time-related dimensions are applicable in many contexts, including e-Business and e-Government. Further dimensions can enrich this base set by taking into account domain-specific requirements. Finally, we have shown that, for some data quality dimensions, there is not yet a general agreement on their definitions in the literature, though the convergence is not far from being reached.

Wand 1996	The ability of an information system to represent every meaningful state of the represented real world system.
Wang 1996	The extent to which data are of sufficient breadth, depth and scope for the task at hand
Redman 1996	The degree to which values are present in a data collection
Jarke 1999	Percentage of the real-world information entered in the sources and/or the data warehouse
Bovee 2001	Deals with information having all required parts of an entity's information present

Fig. 7: Definitions of completeness

References

[Ballou & Pazer 1985] *Ballou, D. P.; Pazer, H. L.:* Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems. Management Science, vol. 31, no. 2, 1985.

[Ballou & Pazer 2003] *Ballou, D. P.; Pazer, H.:* Modeling Completeness versus Consistency Tradeoffs in Information Decision Contexts. IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 1, 2003.

[Ballou et al. 1998] *Ballou, D. P.; Wang, R. Y.; Pazer, H.; Tayi, G. K.:* Modeling Information Manufacturing Systems to Determine Information Product Quality. Management Science, vol. 44, no. 4, 1998.

[Bovee et al. 2001] *Bovee, M.; Srivastava, R. P.; Mak, B. R.:* A Conceptual Framework and Belief-Function Approach to Assessing Overall Information Quality. In: Proceedings of the 6th International Conference on Information Quality (ICIQ 01), Boston, MA, 2001.

[Elfekey et al. 2002] *Elfekey, M.; Vassilios, V.; Elmagarmid, A.:* TAILOR: A Record Linkage Toolbox. IEEE International Conference on Data Engineering '02, San Jose, CA, 2002.

[EU Directive 2003] EU Directive 2003/98/CE on the Reuse of Information in the Public Sector (GUL 345 of the 31.12.2003, pag. 90).

[Fellegi & Holt 1976] *Fellegi, I. P.; Holt D.:* A Systematic Approach to Automatic Edit and Imputation. Journal of the American Statistical Association, vol. 71, 1976.

[Fellegi & Sunter 1969] *Fellegi, I. P.; Sunter, A. B.:* A Theory for Record Linkage. Journal of the American Statistical Association, vol. 64, 1969.

[ISO 2005] *ISO Standard:* ISO/CD TS 1938 Data Quality Measures, (under development), 2005.

[Jarke et al. 1999] *Jarke, M.; Lenzerini, M.; Vassiliou, Y.; Vassiliadis, P.:* Fundamentals of Data Warehouses. Springer-Verlag, 1999.

[Liu & Chi 2002] *Liu, L.; Chi, L.:* Evolutionary Data quality. In: Proceedings of the 7th International Conference on Information Quality (ICIQ 02), Boston, MA, 2002.

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

[Pipino et al. 2002] *Pipino, L. L.; Lee, Y. W.; Wang, R. Y.:* Data Quality Assessment. Communications of the ACM, vol. 45, no. 4, 2002.

[Redman 1996] *Redman, T. C.:* Data Quality for the Information Age. Artech House, 1996.

[Wand & Wang 1996] *Wand, Y.; Wang, R. Y.:* Anchoring Data Quality Dimensions in Ontological Foundations. Communication of the ACM, vol. 39, no. 11, 1996.

[Wang 1998] *Wang, R. Y.:* A Product Perspective on Total Data Quality Management. Communication of the ACM, vol. 41, no. 2, 1998.

[Wang & Madnick 1989] *Wang, R. Y.; Madnick, S.:* The Inter-database Instance Identification Problem in Integrating Autonomous Systems. Proceedings of the 5th International Conference on Data Engineering (ICDE 1989)}, Los Angeles, California, USA, 1989.

[Wang & Strong 1996] *Wang, R. Y.; Strong, D. M.:* Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management Information Systems, vol. 12, no. 4, 1996.

[Winkler 2004] *Winkler, W. E.:* Methods for Evaluating and Creating Data Quality. Information Systems, vol. 29, no. 7, 2004.

Monica Scannapieco is a research associate and a lecturer in the Department of Systems and Computer Science at the University of Rome La Sapienza. Her research interests include data quality models and techniques, cooperative systems for e-government, xml data modeling and querying. She received her PhD in computer engineering from the University of Rome La Sapienza.

Paolo Missier is a research associate at the University of Manchester, UK, since 2004. He has been a research scientist at Telcordia Technologies (formerly Bellcore), NJ, USA from 1994 through 2001, where he gained experience in the area of information management and software architectures. He has also been a lecturer in databases at the University of Milano Bicocca, in Italy, and has contributed to research projects in Europe in the area of information quality management and information extraction from Web sources.

Carlo Batini is full professor of Computer Engineering at University of Milano Bicocca. His research interests include cooperative information systems, conceptual schema repositories and data quality.

Dr. Monica Scannapieco
Università di Roma La Sapienza
Dipartimento di Informatica e Sistemistica
Via Salaria 113 (2nd floor)
00198 Roma, Italy
monscan@dis.uniroma1.it
http://www.disuniroma1.it

Prof. Paolo Missier
University of Manchester
School of Computer Science
Oxford Road
Manchester
M13 9PL, UK
pmissier@cs.man.ac.uk
http://cs.man.ac.uk

Prof. Carlo Batini
Università di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8
20126 Milano, Italy
batini@disco.unimib.it
http://www.disco.unimib.it