Design and Analysis of Quality Information for Data Warehouses*

Manfred A. Jeusfeld[1], Christoph Quix[2], Matthias Jarke [2]

(1) Tilburg University, INFOLAB, Tilburg, The Netherlands
jeusfeld@kub.nl
(2) RWTH Aachen, Informatik V, Aachen, Germany
{quix,jarke}@informatik.rwth-aachen.de

Abstract. Data warehouses are complex systems that have to deliver highly-aggregated, high quality data from heterogeneous sources to decision makers. Due to the dynamic change in the requirements and the environment, data warehouse system rely on meta databases to control their operation and to aid their evolution. In this paper, we present an approach to assess the quality of the data warehouse via a semantically rich model of quality management in a data warehouse. The model allows stakeholders to design abstract quality goals that are translated to executable analysis queries on quality measurements in the data warehouse's meta database. The approach is being implemented using the ConceptBase meta database system.

1 Introduction

Data warehouse systems are ideally suited to investigate the design and analysis of quality information because they are consisting of many components, they involve a large number of stakeholders with different goals, and they are constantly being monitored via administration tools. Figure 1 shows the traditional understanding of data warehouse. They scan by so-called wrappers huge data sets and materialize them in a central (or distributed) database system. Clients for data analysis and decision making access the materialized data sets to generate and validate hypotheses about the enterprise.

Before data warehouses draw the attention of researchers in the early nineties [CGMH+94,KLSS+95,HGMW+95] the integration of heterogeneous data sources was investigated using semantic data models that tried - with limited success - to resolve inhomogeneities to the largest possible extent [Wie92,HZ96]. They concentrated mainly on improving the consistency of the global schema. Data warehouses tackle the problem in a broader way that interestingly makes things easier. Besides consis-

* This work was supported in part by the European Commission in ESPRIT Long Term Research Project 22469 DWQ (Foundations of Data Warehouse Quality).

tency, quality goals like timeliness, accessibility and others [WSF95] come into play which reduce the previously absolute consistency goals to a relative one. Moreover, data warehouses are read-only for the clients and materialize only data of interest to decision making. Via aggregation, errors in individual data sources tend to be minimized by the statistical law of the great numbers.

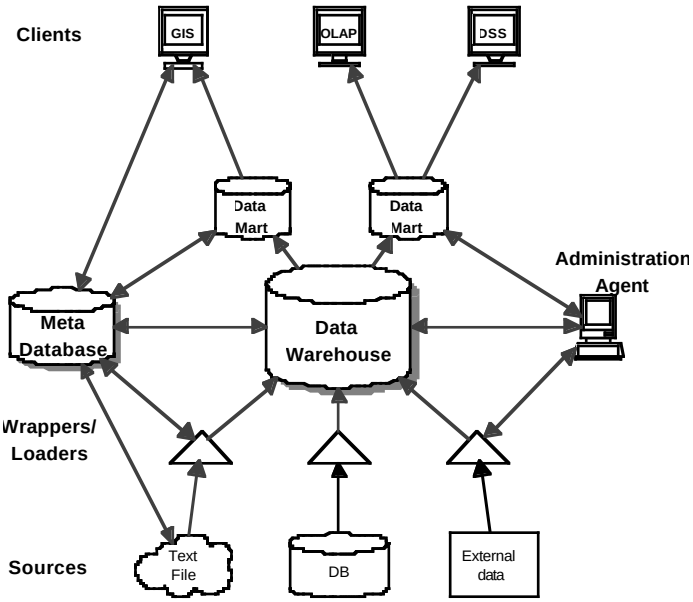

Fig. 1. Current Understanding of a Data Warehouse [JJQ*98]

We claim however that the design and analysis of the quality of a data warehouse is not well-understood and a great problem from the perspective of the users [Jans88]. To tackle the problem, a rich semantic data model was proposed [JJQ*98] for the components of a data warehouse linked to a quality model. The architecture model (see Figure 2) provides for modeling objects at source, data warehouse and client level with perspectives for the conceptual view (a variant of the ER semantic data model), the logical view (relational data model enhanced by aggregation data types), and the physical view (an extension of data flow diagrams). Essentially, we provided in [JJQ*98] a method for representing data warehouse objects in the meta database. Data warehouse objects can come from all levels (source, data warehouse, client), from all perspectives (conceptual, logical, physical), and from any abstraction layer. For example, an instance like a specific object of a table can be represented in the meta database just like its relation definition and even the data model. The same holds for transport agents both at the class level (the transport agent as a program definition) and the instance level (a transport agent process running at a certain time and performing a specific data transport in the data warehouse system).

As a consequence, the meta database of the data warehouse has a rather comprehensive view of the structure of the data warehouse (the class layer) and its current state (the instance layer). The idea of [JJQ*98] was to make all data warehouse objects subject to quality assessment. The problem left open in that paper was how to represent the quality model and how to use it both for quality goal formulation and for quality measurement. Here is where this paper continues.

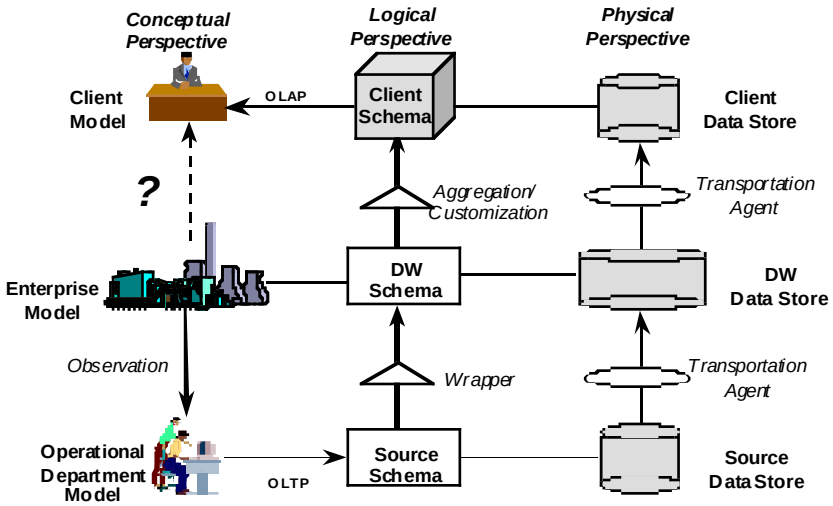

Fig. 2. Levels and perspectives for data warehouse models [JJQ*98]

We argue that the meta database of a data warehouse is the right place to explicitly represent quality goals of stakeholders and to transform them into executable queries on results of quality measurement. The results of quality measurements are also stored in the meta database. Thus, analysis of the quality of a data warehouse shall be done by querying the meta database. Our proposal follows the Goal-Question-Metric approach [OB92] originally developed for software quality management. The main difference in our proposal is representation of 'quality questions' as executable queries on the semantically rich meta database. The meta modeling language Telos [MBJK90] together with the query capabilities of ConceptBase [JGJ*95] form the basis of the presentation throughout the paper. All concepts and queries can be directly inserted into ConceptBase to form a prototype for the data warehouse quality management. In the next section, a collection of important concepts for quality management is discussed along with informal examples. Then, a formal meta model is derived from it and applied to the two tasks of quality goal design and quality analysis. Finally, we discuss limitations of the approach and further work.

2 Basic Concepts for Quality in Data Warehouses

Quality is an issue depending on objects at all levels and perspectives of a data warehouse (figure 2), and depending on quality goals of different stakeholders.

A **measurable object** is an object in the conceptual, logical or physical perspective of a data warehouse. This is the common base class for all classes in the architectural model. Examples are the conceptual schema of a data source, the logical schema of the data warehouse, a wrapper agent, the physical location of a multidimensional data cube and so on. Labeling them measurable object means that some measurable quality goals may be attached to them.

A **quality goal** is an abstract requirement as defined in the Goal-Question-Metric approach of Basili et al. [OB92], it is a related to an object and has a stakeholder, dimension and purpose (see below). As such, a quality goal is a natural-language statement by a stakeholder about her/his quality requirement. It is typically not operational in the sense that there is a unique method to check whether the goal is achieved or not. Examples are ”improve the availability of the source s1 until the end of the month (in the viewpoint of the stakeholder DW administrator)” and ”increase the efficiency of the data loading process (in the viewpoint of the stakeholder source administrator)”. By nature, a quality goal is describing some future state of the data warehouse system that is typically not fulfilled in present time. One may also compare them to user requirements in a software development process. However, in the data warehouse setting, quality goals can be formulated at any time in the life cycle of the data warehouse.

A **quality query** operates on quality measurements to check if a quality goal is currently fulfilled or how the measured quality has changed in a certain period of time. The quality query does not only compare expected range with the achieved value of the quality measurement, it can have a more complicated method to check if the quality goal is fulfilled or not (e.g. by the combination of several quality measurements). An example query would be whether the availability value of data source S1 is in the expected range of 97%. Quality queries are the formal (and executable) specifications on how to check whether the quality goal for which it provides evidence is achieved or not. In our model, we assume that any information necessary to decide that is in the DW meta database. Thus, a query to the meta database is sufficient to evaluate a quality query. The algorithms for quality measurements are hidden in the metric agents (see below): they store a quality value as result of a measurement in the meta database.

A **quality dimension** is a term used to formulate quality goals. The terms are arranged in a specialization hierarchy. Each stakeholder may have his/her own hierarchy of preferred quality dimensions. Examples are data usage quality, accessibility, consistency, timeliness etc. The difference of quality dimension to quality goal is that the latter has a) a direction like ‘improve’ or ‘achieve’, and b) a measurable object it is

applied to. Quality dimensions provide the vocabulary to formulate quality goals. Each stakeholder may have a different vocabulary, i.e. hierarchy of quality dimensions.

A **quality measurement** is the documented activity to measure the quality of some measurable objects. The documentation establishes a static relationship between a measurable object and the measured quality value. It uses a quality metric to get an actual quality value of the object, it has an interval of expected values for the quality value, and it is related to a quality dimension. Quality measurements may be specialized for measuring a specific kind of DW objects, e.g. measurement of Non-Null-Tuples applies only to relations being part of the logical schema. An example would be like ”the current value for availability of source s1 is 23.5 hours per day”.

A **metric unit** specifies the dimension of a quality value. It is analog to physical units like ”meter/second” for measuring speed. We allow that for a given kind of quality measurement, multiple units are possible. For example, one could have ”nullvalues / tuples” or ”nullvalues / attributes” as metric units for measuring the completeness of a relation. Like for physical units, there can be an algebra for transforming a quality value from one metric unit to another.

A **quality domain** specifies permissible quality values which themselves are the results of measurements. Quality values are instances of a quality domain. Domains should be totally ordered, i.e. for two values x,y of a domain either $x < y$, or $x=y$ or $y<x$ holds. Quality values are different from quality measurements: a measurement is the activity to attach a quality value to a measurable object. Examples are 23 hours and 95%. The pure value has no meaning by itself. Only the link to a measurable object and (via a quality query) the link to a quality goal establishes the context necessary to interpret such a value!

A **quality range** is a set of quality values, in particular of **expected** quality values. The most common form of ranges are intervals. We generalize this to subsets of quality domains. Finally, a **stakeholder** is a person who formulates quality goals for the data warehouse. Typically, a stakeholder receives as feedback answers to the quality queries which provide evidence for his/her quality goals. The DW administrator, the decision maker and the source administrator are all stakeholders with different quality goals.

3 The Quality Meta Model

The above glossary could used be used to understand the issues of quality management. By using a meta modeling approach, it can be made part of the meta database schema of the data warehouse. By doing so, the stakeholders can represent their quality goals explicitly and the meta database maintains the relationship to quality values of measurable objects. Thus, the quality meta model can be used for both design and analysis purposes. To do so, one has to take into account that a quality measurement

maps an arbitrary object of a data warehouse (e.g., the number of null values of source relation 'Sales') to some value. Thus, a quality measurements relates concepts with different abstraction levels, here a schema concept to a number.

Because of this, we propose a quality meta model (see Figure 3) whose concepts are all at the meta class level (indicated by light gray boxes) with the notable exception of 'MeasurableObject' being a meta meta class (white box). The quality meta model provides the notation for formulating quality goals, queries, and measurements. The upper part around 'QualityGoal' allows stakeholders to formulate quality requirements for 'MeasurableObject'. Typical instances of 'MeasurableObject' are 'Relation', 'Agent' etc., i.e. all kinds of objects present in a data warehouse and represented in its meta database. The purpose of a quality goal is interpreted as a direction, e.g. 'increasing' or 'achieving' some quality goal. It can be used for mapping the quality goal into a query. The description is just a string like the examples for quality goals in the previous section. Finally, a quality goal is linked to the quality dimensions, e.g. availability.

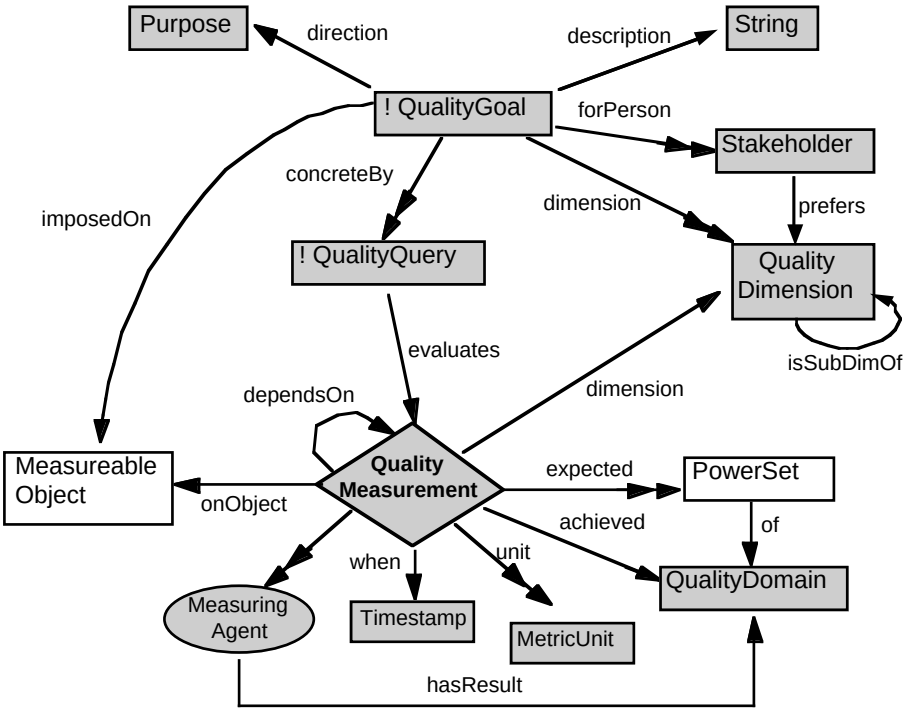

Fig. 3. The quality meta model

While the upper third around quality goal is abstract and design-oriented, the lower third is concrete and analysis-oriented. A measurable object is linked to a quality value

by a quality measurement, not incidentally displayed like an ER relationship type. The quality measurement also links a measurable object to expected an unexpected ranges for the quality values. It can be compared to a temperature meter showing a certain value on a scale with green and red ranges. By attaching such a quality measurement explicitly to a measurable object, we express that we want to monitor the quality of that object.

It is essential that MeasurableObject is at a different abstract level. Just consider a concept like the relation 'SalesEurope'. It is at the abstraction level of a simple class whose instances are the tuples of the relation. However, a quality value on the availability of this relation is just a number, i.e. an instance object. In functional terms, a quality measurement maps a concept (like SalesEurope) to a concrete number.

In our current model, quality dimensions are related to stakeholders and they form a subsumption hierarchy, e.g. traceability is sub-dimension of Design & administration quality (DW Designer /Administrator viewpoint). We want to stress that the hierarchy of quality dimensions depends on the stakeholder, too. Every stakeholder prefers its own quality dimensions and arranges them in her own hierarchy. In ConceptBase, different viewpoints (roughly hierarchies of quality dimensions) can implemented in different modules, so that the viewpoints are independent from each other.

The analysis of these goals are supported by queries working on quality measurements. The relationship between quality queries and quality measurements will be mostly defined implicit through the query classes of ConceptBase.

4 Specialization and Instantiation of the Model

When we instantiate quality goals, we encode actual quality goals of stakeholders like shown above. Instances of quality measurements encode the plan to measure an instance of DW_Object, e.g. the relation 'SalesEurope' for its quality value on availability. The figure below defines a quality goal 'AvailGoalforRel' that states that stakeholders 'DW Administrator' may in principle be interested in achieving a certain level of availability for (source) relations.

As mentioned above, the natural language definition of quality goal is taken from the GQM model. Goals are represented there as a template of purpose, issue (called quality dimension here), object and viewpoint (called stakeholder here). Consider the following two examples of quality goals:

Purpose:	Achieve
QualityDimension:	Availability
Measurable Object:	source relation SalesEurope
Stakeholder:	Data Warehouse Administrator
Description:	"... at least once per week"

Purpose: Increase
QualityDimension: Efficiency
Measurable Object: process data loading
Stakeholder: Source administrator
Description: "... increase speed of data loading by 25%"

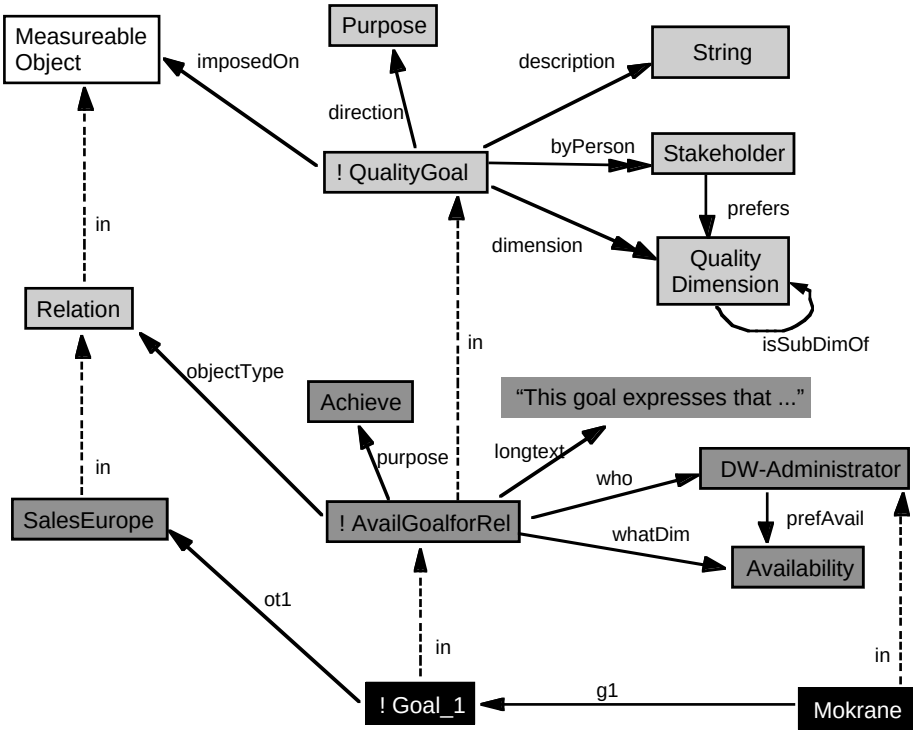

Fig. 4. Use of the quality model for goal formulation

Figure 4 shows how quality goals are encoded using the quality meta model. The dark gray boxes are simple classes (schema level). They are pattern to express instance quality goals like 'Goal_1' of stakeholder 'Mokrane'. The figure is not a one-to-one translation of the above textual examples. Instead, one part of the example goes to the simple class level (gray boxes) , and the other to the instance level (black boxes). At the simple class level, a goal 'AvailGoalforRel' is formulated that refers to object type 'Relation'. The purpose is set to 'Achieve' and a description is given. Moreover, it is stated that the DW Administrator is in principle interested in such a goal. At the instance level, 'Goal 1' represents the fact that 'Mokrane' (being a real DW administrator) has instantiated this goal for the (source) relation 'SalesEurope'. Note the different abstraction levels of the 'Goal 1' and 'SalesEurope'! By simple instantiation, the same quality goal can be attached/detached to multiple data warehouse objects, here

relations. The middle layer of figure 4 represents patterns of quality goals rather than actual quality goals which are found in the lower layer. Thus, the middle layer constitutes re-usable quality goals that only have to be parameterized by the object type (here: an instance 'Relation') and the stakeholder (here an instance of 'DW-Administrator'). A company using our approach may populate this middle layer by their specific collection of quality goal patterns. This collection encodes the *domain knowledge* of that company about formulating quality goals.

Figure 5 instantiates the quality measurement part of the meta model. Again, it defines a pattern for measurements at the simple class level. Here, the class 'measureNullValues' is defined to measure relations into percentages of null values per tuple (100% means that all values of all tuples are NULL). The exact interpretation is hidden in the description of the metric unit '#nullvalues/#tuples'. A range '[0;2]' is defined as the expected interval The metric agent is 'nv_counter' which possibly only accesses a small portion of the relation to estimate the achieved quality value.

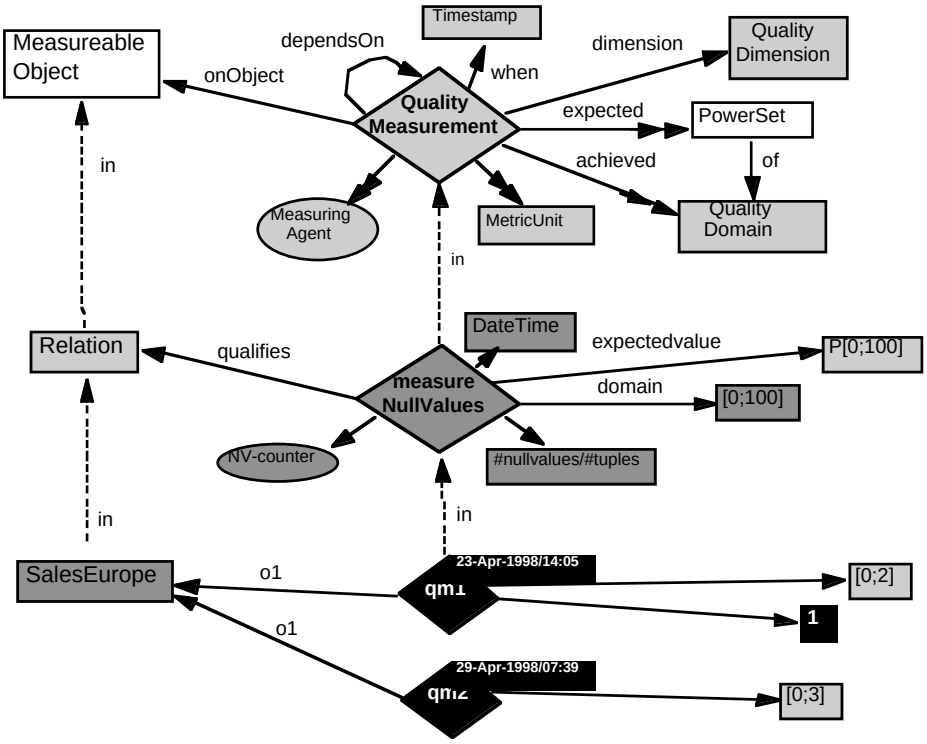

Fig. 5. Use of the quality model for measurements

The definition of the expected range as a query class in ConceptBase is done by queries defining subsets out of the possible quality domain.

```
QueryClass [0;2] isA [0;100] with
  constraint
    c: $ (this >= 0.0) and (this =< 2.0) $
end


Class [0;100] in QualityDomain isA Real with
  constraint
    c: $ (this >= 0.0) and (this =< 100.0) $
end
```

A quality domain is defined by an ordinary class with membership constraint. Note that query classes automatically classify the instances of their super class which fulfill the constraint of the query class.

At the instance level, two measurements are shown. The first 'qm1' results in an expected quality value, the second is an incomplete instance that represents a future measurement. Because the ranges are formally defined as queries, the measured quality values are automatically classified to be in the expected or unexpected range. The expected range is attached at the instance level. Thereby, each quality measurement can have its own expected range. Like measured quality values, the expected ranges may change over time. A partial instance like 'qm2' is interpreted as the plan to do the measurement of the specified class on the specified object at the specified time expecting the specified range.

What remains open is the role of the quality query in the meta model. It is placed between quality goal and quality measurement. The purpose of quality query is to mediate between the quality goal (an abstract requirement that cannot be assessed directly) and a measurement (yielding a concrete quality value).

```
QualityQuery TooManyNullValues isA Source,Relation with
  constraint
     c: $ exists m/MeasureNullValues
           (this hasMeasure m) and
           not (m in MeasureNullValues^exprange) $
end
```

That query is *evidence* for a quality goal like 'AchieveCompl' (achieve completeness of source relations). Note that the query restricts the relations to be measured. A more complicated query has to be defined for the purpose of increasing a certain quality goal, e.g. increase the number of source relations which have an expected quality value for the percentage of null values. The instances of the query 'BetterOnNullValues' shown below are all source relations of the data warehouse which perform better on the null values (based on two measurements). The attribute 'after' was not shown in the meta

model to ensure the readability. The objects ‘Source’ and ‘Relation’ locate the kind of data warehouse objects that are of interest to this query: it is the source schema of the logical perspective in figure 2.

```
QualityQuery BetterOnNullvalues isA Source,Relation
with constraint
     c: $ exists m1,m2/MeasureNullValues
            (this hasMeasure m1)  and
            (this hasMeasure m2) and
            (m2 after m1) and
            exists v1,v2/[0;100]
              (m1 qualityvalue v1) and
              (m2 qualityvalue v2)
       ==> (v1 > v2) $
end
```

In the implementation of the quality model, a metric agent would store the precedence of measurements in the meta database. More detailed information about measurements is also possible. For example, the attribute ”when: Timestamp” specifies for a quality measurement when it was (or will be taken). Then, an instance like ‘qm2’ could lack a quality value but could have a ‘when’ attribute sometime in the future. Such incomplete measurement objects denote a schedule of measurements very much like a crontab file in UNIX.

The quality meta model of figure 3, a quality goal is made concrete by multiple quality queries. Quality queries themselves accesses quality measurements attached to data warehouse objects. Data warehouse objects are classified in the DW meta database according to level, perspective, and abstraction layer as introduced in section 1. Hence, quality queries aggregate information of rather heterogeneous nature in order to provide evidence for the fulfillment or non-fulfillment of a quality goal.

The usefulness of the quality queries depends on the completeness, timeliness, accuracy, and consistency of the meta database, particularly of the quality measurements materialized there. The meta database itself can be regarded as a view on the data warehouse system which has to be maintained according to *secondary* quality goals. We will however not elaborate on this problem here.

5 Conclusions

We presented a quality meta model for data warehouses that can be used for both design of quality and analysis of quality measurements. The model can be directly incorporated into the meta database system ConceptBase. The main advantages are:

- Any kind of measurable object is allowed as long it is represented in the meta database of the data warehouse. Specifically, static objects like schema concepts and dynamic objects like wrappers are supported.
- Quality goals can be formulated from the perspectives of an extensible set of stakeholders. Each stakeholder can assess the data warehouse quality from his/her perspective by evaluating the quality queries attached to his/her quality goals.
- Quality queries are executable queries on the meta database. Their answers are the evidence for a stakeholder to decide whether the quality is appropriate or not. At any time quality queries can be inserted, extended, modified, and removed. This is due to the fact that the meta database is not just a CASE repository but an integral part of the runtime data warehouse system.
- Quality measurements are explicitly stored in the meta database. By materializing sequences of quality measurements of the same type in the meta database, one can realize more advanced quality goals about trends by appropriate quality queries.

Besides the integrated quality meta model, a major achievement is the separation of quality information at the instance and schema level. The latter provides for a collection of reusable quality goals and measurements that are activated by simple instantiation in the meta database.

There are a couple of research questions to be addressed. First, a suitable collection of quality metrics for data warehouses has to be investigated. Starting point is the research and practice on metrics in the software development area. Second, a suitable strategy for materializing the quality measurements is missing. For the moment, we assume that there are external metric agents that compute some quality value for a given measurable object. But when should the agent be activated? Supposedly, measuring the quality of a component like a source relation is computationally expensive. One simply cannot afford to measure the quality for all components continuously. Interestingly, the quality of the materialized quality measurements can be assessed like the quality of any other component. They have a certain accuracy, a certain timeliness etc.

The interdependencies between quality measurements has not been addressed in this paper. One can assume that the quality of interacting data warehouse components depend on each other. For example, the completeness of a materialized data cube depends on the completeness of the data sources. Would it be possible to estimate the

quality value of the data cube based on the quality values of the data sources, or vice versa?

The approach is currently implemented in the Esprit Project Foundations of Data Warehouse Quality (DWQ). Test beds are being provided by associated partners from the telecommunications industry and the insurance business. A prototype showing the design and analysis capabilities is finished and being demonstrated at data warehouse conferences and workshops [GJJ*98].

Research on quality management for data warehouses is still at its genesis. Data warehouses are an excellent research object in this respect: there has been little investigated so far and the need for a solution is imminent. The quality meta model elaborated on in this paper provides the structure for representing knowledge on how to do the quality management for a given application. The actual selection of quality goal patterns and quality measurement plans is subject to the application domain. Thus, an important next step is to populate the meta database with domain knowledge on how to organize quality management for a given application.

A further research goal is to extend to method to the design of a data warehouse which includes view refreshment strategies, selection of the right source databases, filters, transport agents etc. based on their quality properties. Our approach currently concentrates on the assessment of the quality of a given data warehouse solution. By linking it to a quality-oriented design/evolution method, we hope to establish a feedback cycle for continuous improvement of a data warehouse solution according to changing quality goals.

Acknowledgments: Many thanks go to our DWQ partners, especially to Mokrane Bouzeghoub, Maurizio Lenzerini, Panos Vassiliadis, and Enrico Franconi for commenting on earlier versions on the quality meta model.

6 References

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom. The TSIMMIS project: integration of heterogeneous information sources. In *Proc. of IPSI Conference*, Tokyo (Japan), 1994.

[GJJ*98] M. Gebhardt, M. Jarke, M.A. Jeusfeld, C. Quix, S. Sklorz. Tools for data warehouse quality. In Proc. 10th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'98), Capri, Italy, July 1-3, 1998.

[HGMW+95] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, Y. Zhuge. The Stanford Data Warehousing Project. *Data Eng., Special Issue Materialized Views on Data Warehousing*, 18(2), pp. 41-48, 1995.

[HZ96] R. Hull, G. Zhou. A Framework for supporting data integration using the materialized and virtual approaches. *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pp. 481–492, Montreal, 1996.

[Jans88] M. Janson. Data quality: the Achilles heel of end-user computing, *Omega J. Management Science*, 16, 5, 1988.

[JGJ+95] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, S. Eherer. ConceptBase - a deductive object base for meta data management. In *Journal of Intelligent Information Systems*, 4, 2, pp. 167-192, 1995.

[JJQ*98] M. Jarke, M.A. Jeusfeld, C. Quix, P. Vassiliadis: Architecture and quality in data warehouses. In *Proc. 10th Intl. Conf. CAiSE*98*, Pisa, Italy, June 8-12, 1998, pp. 93-113, Springer-Verlag, Berlin.

[JV97] M. Jarke, Y. Vassiliou. Foundations of data warehouse quality -- a review of the DWQ project. *Proc. 2nd Intl. Conf. Information Quality (IQ-97)*, Cambridge, Mass., 1997.

[KLSS95] T. Kirk, A.Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. *Proc. AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, pp. 85-91, 1995.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. Telos – a language for representing knowledge about information systems.. In *ACM Trans. Information Systems*, 8, 4, pp. 325-362, 1990.

[OB92] M. Oivo, V. Basili. Representing software engineering models: the TAME goal-oriented approach. *IEEE Trans. Software Eng*. 18, 10, 1992.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pp. 38-49, March 1992.

[WRK95] R.Y. Wang, M.P. Reddy, H.B. Kon. Towards quality data: an attribute-based approach, *Decision Support Systems*, 13, 1995.

[WSF95] R.Y. Wang, V.C. Storey, C.P. Firth. A framework for analysis of data quality research, *IEEE Trans. Knowledge and Data Eng.* 7, 4, 1995.