Managing Quality Properties in a ROLAP Environment

Adriana Marotta[1], Federico Piedrabuena[1], Alberto Abelló[2]

[1]Instituto de Computación, Universidad de la República, Montevideo, Uruguay
[2] Universitat Politècnica de Catalunya, Barcelona, España
[1]{amarotta, fpiedrab}@fing.edu.uy, [2] aabello@lsi.upc.edu

Abstract. In this work we propose, for an environment where multidimensional queries are made over multiple Data Marts, techniques for providing the user with quality information about the retrieved data. This meta-information behaves as an added value over the obtained information or as an additional element to take into account during the proposition of the queries. The quality properties considered are freshness, availability and accuracy. We provide a set of formulas that allow estimating or calculating the values of these properties, for the result of any multidimensional operation of a predefined basic set.

1 Introduction

The use of OLAP systems has become common practice, and in current times enterprise managers have the possibility to analyze the whole enterprise information with a multidimensional visualization. This information is obtained querying many databases, whose data is prepared for this kind of analysis. These databases are called Data Marts (DM). Each DM contains information that is represented in a multidimensional manner, is oriented to certain subject of interest, and comes from other databases, such as enterprise Data Warehouses or operational databases.

The user, who receives the answers of the queries submitted to the DMs, is quite far from the generation of the information (unknowing how and when it was generated at sources and loaded to DMs), therefore it would be extremely useful for him to count with additional information. This meta-information would make him feel more confident about the decisions he is making based on the information retrieved by the queries and, on the other hand, would allow him to eventually reformulate his queries.

Data quality is being deeply studied since some years ago. There exists various quality properties defined in the literature, such as *completeness, accuracy, accessibility, freshness, availability*. In this direction, there is much work for the particular case of information systems that are fed from multiple sources [1] [2] [3]. The possibility to provide a considerable amount of quality information to the users of such systems, and its importance, is globally recognized.

The meta-information associated to the retrieved information from the DMs may consist of a set of values corresponding to certain quality properties. We consider *freshness, availability and accuracy* constitute an interesting group of quality properties for this context, due to functional characteristics of the DMs and intrinsic characteristics of the information received by the user. On one hand, each DM is periodically maintained, which generates constraints in the DM availability and restricts the freshness of its data. On the other hand, the information obtained by the user is the result of combinations of data coming from different sources, with probably heterogeneous levels of accuracy, thus the accuracy of this information is not a trivial meta-information and is a significant added value for the user.

As an example, consider a director of a multi-national enterprise, who needs information about its sales throughout several countries. For obtaining it, he makes queries over DMs belonging to the different countries' subsidiaries. These DMs follow certain standards of formats, codification, etc., so that their information can be integrated by multidimensional operations. Suppose the director obtains sales amounts corresponding to South-America, discriminated by products, for each month of the current year. We believe that this person would be really grateful if he also obtained meta-information such as when the involved data was loaded in the DMs and how accurate it is. And still more, if previous to the execution of the query he is informed about the availability of the data he is requiring, and alternative DMs for obtaining the non-available information are suggested to him.

Some works about federations of OLAP systems and other technologies can be found in the literature [4] [5]. This work is situated in the context of an environment where multidimensional queries are made over multiple DMs, sometimes combining various DMs in the same query. Our goal is to propose techniques for giving support to users, providing them quality information about the information retrieved, as an added value over the obtained information or as an additional element to consider during the proposition of the queries.

Given a query over DMs, proposed by the user, and the freshness, availability and accuracy values of each DM, the problem we address consists on calculating: (i) availability of the needed DMs' information, and (ii) freshness and accuracy of the information resulting from the executed query. The techniques we propose use existing proposals for quality properties propagation in Relational Algebra operations [1][6][7] as a reference point. Nevertheless, they involve particular criteria that are related with a multidimensional classification of the relational elements and other particularities of the OLAP context.

The contribution of this work is the proposal of: (i) a specific scenario where OLAP-queries results are enriched with quality meta-information, and (ii) techniques for calculating the availability, freshness and accuracy values of information that was obtained through application of OLAP queries to multiple DMs.

The rest of the paper is organized as follows. Section 2 presents the context of the work, Section 3 presents the quality properties we use, in Section 4 we present our proposal for quality evaluation, and finally in Section 5 we present the conclusions.

2 Context

We establish here a concrete scenario, for which we propose the quality management techniques. First, we describe the basic scenario that we consider as starting point for our work. After, we present the enriched scenario, which is the basic one enriched with our proposal.

2.1 Basic Scenario

We consider a ROLAP system (OLAP over RDBMS), where users execute multidimensional queries. These queries are composed by multidimensional operations. In this section we present the scenario and the multidimensional-operations set considered.

OLAP functionality is characterized by dynamic multidimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities (interactively exploring cubes). In an OLAP system each multidimensional operation is a function that takes Cubes (a set of cells placed in an n-dimensional space) as arguments and returns a Cube. "ROLAP" tools automatically generate a SQL query according to the operations performed by the user, see Figure 1. Many times end users navigate from Cube to Cube not just applying isolated operations but performing sequences of operations; this sequence is performed as a sequence of SQL queries.

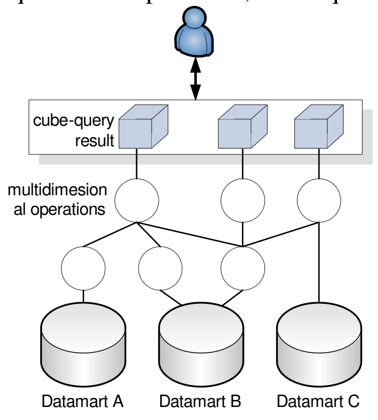


Fig. 1. ROLAP System.

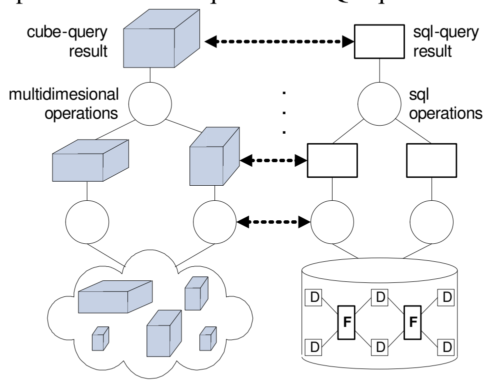


Fig. 2. Relational implementation of multidimensional concepts.

Given that it does not exist any standard multidimensional algebra or data model, we use YAM^2 [9] multidimensional algebra for the set of operations, because it provides a direct translation to SQL and it allows making any multidimensional query; in [8] this algebra is stated as complete, which assures us the expressiveness offered. Multidimensional operations are represented as SQL queries using a cube-query template that was proposed in [8] for retrieving a Cell of data that conforms to a Cube, from the RDBMS. Figure 2 shows the correspondences between an OLAP and a ROLAP system, each multidimensional operation can be translated to a cube-query and each cube can be translated to a relation.

Each DM in this scenario is a set of cubes that correspond to the same subject of analysis, i.e. to the same fact. These cubes are represented by the Relational Model, through the star (see Figure 3), snowflake (see Figure 4) or a hybrid schema. Completely normalizing each Dimension we get a snowflake schema and not normalizing them at all results in a star schema. We choose the generic approach, like in [10]. With respect to the Fact, it is defined as a set of Cells, which are materializations of different levels of aggregations of the same fact. The Dimension Levels may be materialized according to the materializations of the Cells, since the latter have FKs that must point to the corresponding PKs in the Level relations.

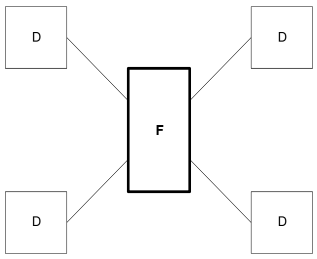


Fig. 3. Star schema.

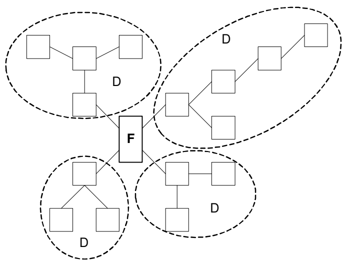


Fig. 4. Snowflake schema.

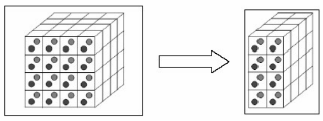
We define two kinds of DMs, according to the volatility of their data, and we assume that our system is organized this way: (1) Stable DM, contains only historical information, whose fact-dates range between two certain dates in the past. Therefore, this DM is not refreshed nor loaded any more. (2) Non-stable DM, contains information that comes up to the present date. This DM is periodically refreshed.

Multidimensional Operations. We use the SQL template (Cube-query), for constructing the different multidimensional operations.

Cube-query:
```
RESULT =  SELECT l1.ID, …, ln.ID, c.Measure1, …
          FROM Cell c, Level1 l1, …, Leveln ln
          WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
          GROUP BY l1.ID, …, ln.ID
          ORDER BY l1.ID, …, ln.ID
```

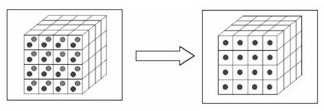
The following are the intuitive operations definitions and the associated SQL code.

Dice. By means of a predicate P over a Dimension attribute, this operation allows to choose the subset of points of interest out of the whole n-dimensional space.



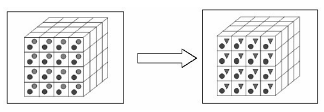
```
SELECT l1.ID, …, ln.ID, c.Measure1, …
FROM Cell c, Level1 l1, …, Leveln ln
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
AND P
GROUP BY l1.ID, …, ln.ID
ORDER BY l1.ID, …, ln.ID
```

Projection. This just selects a subset of Measures from those available in the Cube.



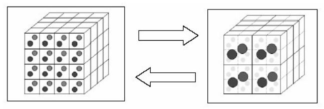
```
SELECT l1.ID, …, ln.ID, c.Measure1, …
FROM Cell c, Level1 l1, …, Leveln ln
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
GROUP BY l1.ID, …, ln.ID
ORDER BY l1.ID, …, ln.ID
```

Drill-across. This operation changes the image set of the Cube. The n-dimensional space remains exactly the same, only the cells placed in it change.



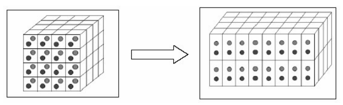
```
SELECT l1.ID, …, ln.ID, c.Measure1, …,
c'.Measure1', …
FROM Cell c,Cell' c', Level1 l1, …, Leveln ln
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
AND c'.key1 = l1.ID AND … AND c'.keyn = ln.ID
GROUP BY l1.ID, …, ln.ID
ORDER BY l1.ID, …, ln.ID
```

Roll-Up. It groups cells in the Cube based on an aggregation hierarchy, modifying the granularity of data. Assuming the two levels are materialized separately, suppose level$_1$ and level$_k$ are two contiguous levels from the same dimension (it exists a foreign key from level$_1$ to level$_k$), and we want to roll up from level l$_1$ to level l$_k$.



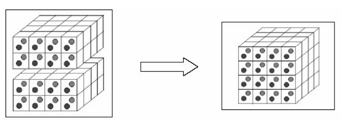
```
SELECT lk.ID, …, ln.ID, F(c.Measure1), …
FROM Cell c,Level1 l1, …, Leveln ln, Levelk lk
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
AND l1.keyk = lk.ID
GROUP BY lk.ID, …, ln.ID
ORDER BY lk.ID, …, ln.ID
```

Change Base. This operation reallocates exactly the same instances of a Cell in a new n-dimensional space with exactly the same number of points. Thus, it actually modifies the analysis dimensions used.



```
SELECT n1.ID, …, nn.ID, c.Measure1, …
FROM   Cell   c,   Level1   l1,   …,   Leveln   ln,
NewLevel1 n1, …, NewLeveln nn
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
AND l1.att1 = n1.ID AND … AND ln.attn = nn.ID
GROUP BY n1.ID, …, nn.ID
ORDER BY n1.ID, …, nn.ID
```

Union. We propose a variant of Union operation proposed in [8], because, is very important to consider the possibility of combining cubes, which we know they have the same schema and formats, but we do not know anything about the data they contain. This operation performs the union of two n-dimensional identical cubes.



```
SELECT l1.ID, …, ln.ID, c.Measure1, …
FROM Cell c, Level1 l1, …, Leveln ln
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
GROUP BY l1.ID, …, ln.ID
ORDER BY l1.ID, …, ln.ID
UNION
SELECT l1.ID, …, ln.ID, c.Measure1, …
FROM Cell' c, Level1' l1, …, Leveln' ln
WHERE c.key1 = l1.ID AND … AND c.keyn = ln.ID
GROUP BY l1.ID, …, ln.ID
ORDER BY l1.ID, …, ln.ID
```

2.2 Enriched Scenario

We enrich the scenario presented in last section with the quality evaluation of the queries that are posed by the user. In the following, we present the way it would work. The user of the OLAP system poses an OLAP query over various DMs. This query is decomposed into various multidimensional operations and translated to SQL. Availability is calculated through the sequence of operations, knowing if the information required by the user is available or not. The user is informed about it. Suppose it is available. Next, the user asks the system for the estimations of freshness and accuracy of the cube he will obtain. The system gives him a freshness value of the cube and an accuracy value for each of the cube measures he will obtain. The user decides to change a bit his query, excluding one of the participating DMs, and the estimated values improve. Finally, the user executes the query, obtaining the cube with its associated quality metadata, which was exactly calculated by the system. This metadata includes, the cube freshness value, the accuracy values for each cube measure, and also includes these values for each tuple that can be seen by the user if he is interested in going more in detail.

System Metadata. To provide a comprehensive picture of the overall ROLAP system we use an extension of the Federation Ontology for Multi-Source Information System (MSIS) stated in [11]. It provides six metadata categories of federation information (e.g. quality properties indicators, access control directives topology information), which are highly flexible. In this context we only need two categories:

- Data Quality: This category defines the data quality properties of the DM data and the multidimensional data result.
- Source Quality of Service: This category defines the quality properties of the service of each DM. The DM resolves each multidimensional operation by means of a process or service.

Each DM provides its own metadata to be used by the OLAP system.

3 Quality Properties

The quality properties we have chosen for our scenario are *freshness, availability and accuracy*. This choice is mainly due to two reasons: the usefulness that they may have for the DM users, and the existence of previous work about them that can be applied to our context. Based on existing approaches and definitions of these properties we propose one specific definition for each one, which we think are the most suitable for our needs. Besides each definition we state the level of granularity at which we apply the property, e.g. a whole DM, a cube, a table, each tuple, etc.

3.1 Freshness

Several definitions can be found in the literature for quality properties related to the age of the information. Some of them can be found in [12][13][1], while in [14] they present a summary and classification of a wide set of existing approaches for this property. For this work we propose the definition: *Freshness is the time elapsed since the data was loaded in the DM until it is received by the user through a query.*

We propose for the assignment of freshness values to the system, a DM granularity, a cube granularity and in some cases a tuple granularity. At each DM we do not manage tuple granularity, because we assume it is entirely loaded on a periodic basis, therefore it would be useless to have the possibility of different freshness values in different portions of the DM.

The metadata needed for freshness information at the DM is the following:

- Loading Period (lp): An integer that represents the quantity of hours passed between the starting time of two loading processes (period between loadings).
- Loading Duration (ldur): An integer that represents an approximation of the quantity of hours that passes between the starting time of the loading process (the downing of the DM service) and its ending time (the starting service instant).
- Last Loading Date/Time (lastl): A decimal that represents the date and time when the last loading process was started.
- Stable-or-Not (st): A boolean that indicates if the DM is stable, i.e. if the DM is not refreshed because its information should cover just until certain date (as we explained in Section 2.1).

In the case the DM is stable:

$$\text{lp} = \text{ldur} = 0\ . \tag{1}$$
$$\text{lastl} = \text{NOW}\ . \tag{2}$$

where NOW is a special value, which means the actual moment.

The metadata needed for freshness information in the whole system is the following:
For each fact table (cube):

- Freshness Value (fv): An integer that represents the quantity of hours elapsed since the table data was loaded in the DM.

For each fact table, for each tuple:

- Freshness Value by Tuple (fv_by_tuple): An integer that represents the quantity of hours elapsed since the tuple was loaded in the DM.

The freshness information at the DM allows us to calculate the exact value of freshness at a certain moment at the DM (fv), and also the maximum and minimum freshness values (maxfv and minfv) that are possible at the DM.
We calculate these DM values through the following formulas:

$$\text{fv} = \text{round}(\text{actual\_datetime} - \text{lastl})\ . \tag{3}$$
$$\text{maxfv} = \text{lp} + \text{ldur}\ . \tag{4}$$
$$\text{minfv} = \text{ldur}\ . \tag{5}$$

Note that for stable DMs:

$$\text{fv} = \text{maxfv} = \text{minfv} = 0\ . \tag{6}$$

Each fv_by_tuple of each DM fact table is set to the fv of the DM.

3.2 Accuracy

The term accuracy is frequently used to describe many different aspects related to the data itself. Semantic correctness, syntactic correctness, precision and fuzziness are some of the concepts related to accuracy that are used with similar intentions [1], [6], [15], [16]. For this work we propose the definition: *Accuracy is the probability of an attribute value to be correct. A value is correct if corresponds to reality.*

The granularity we use for this property is at attribute level, but we manage accuracy information only for the measure attributes. This is because we consider this kind of attributes is the most relevant for the OLAP analysis, and also this restriction allows us to find solutions that are more specific to certain type of information. For each measure attribute we manage DM, cube and, sometimes, tuple granularity. At each DM we do not manage tuple granularity, because the measurement of accuracy at the DMs could not be made for each value of the attributes, since they are numeric values and cannot be easily verified (considering errors generated by wrong digitations). For example, it is not possible to use tools such as dictionaries or look-up tables, which allow verifying string values belonging to determined domains. Therefore, we suppose that the accuracy values at the DMs are estimated through statistic methods or by people involved in the generation of the data.

The metadata needed for accuracy information is the following:
For each fact table, for each measure attribute:

- Accuracy Value (av): A decimal number between 0 and 1 that represents the probability of being correct for the values of the attribute.

For each fact table, for each measure attribute, for each tuple:

- Accuracy Value by Tuple (av_by_tuple): A decimal number between 0 and 1 that represents the probability of being correct for the value.

At the DMs, each av_by_tuple of each measure of the fact tables is set to the av of the measure attribute.

3.3 Availability

Availability is a measure of a system or service readiness to perform its function when it is needed [17], [12]. For this work we propose the definition: *Availability indicates if a service is ready for use at a given instant.*

We assume that the only factor that influences the availability of the DM is the loading process and that the DM is totally unavailable during this process. We manage a DM granularity and in the rest of the system a cube granularity.

The metadata needed for availability information at the DM is the same as for freshness: i) loading period (lp), ii) loading duration (ldur), iii) last loading date/time (lastl) and iv) stable-or-not (st).

The metadata needed for availability information in the whole system is the following:

- Avalilability Value (vv): A Boolean that represents if the DM is available (vv = TRUE) or not (vv = FALSE).

The metadata at the DM allows us to calculate the availability of the DM at a certain moment (vv). We calculate this value through the following formula:

$$vv = actual\_datetime >= (lastl + ldur) . \quad \textbf{(7)}$$

Note that for stable DMs:

$$vv = true(due\ to\ lastl = actual\_datetime\ and\ ldur = 0) . \quad \textbf{(8)}$$

4 Quality Evaluation of User Queries' Results

Our goal is to determine how to obtain the quality values of the information retrieved by a query in our defined context. We propose techniques for two kinds of quality evaluation: (i) quality values calculation, after the query is executed, and (ii) quality values estimation, before the query is executed. For the calculation, we need to manage a granularity of tuple, since it takes into account the resulting tuples of each operation. In contrast, for the estimation we manage a cube granularity, since we do not need the information about the obtained tuples in each operation.

We provide a formula for each quality property and operation. For a sequence of operations, the formulas are composed to obtain the quality values of the final cube. The estimations and calculations are done taking into account the SQL specifications for each multidimensional operation given in Section 2.1. The formulas we propose are based on previously proposed formulas for the Relational Algebra (RA) operators [1] [6] [7]. However, they take into account three main aspects that characterize our context: (i) the type of the elements that participate in each operation. Different types of relational elements are distinguished in the operations specifications. A relation can be a *Cell* (or fact table) or a *Level* (a table containing dimension data). An attribute can be a *Measure*, the key of a dimension level or some other dimension attribute. (ii) the relevance that each type of element has in the resulting quality. We state that the quality of the fact tables determines the quality of the user queries results. Therefore the fact tables and measure attributes must have the greatest influence in the quality evaluation formulas. (iii) the granularity managed. The OLAP context particularities suggest us certain granularities for the quality values, which are suitable and useful. We consider the cube granularity because each multidimensional operation result is a cube (in particular the user query result). We also work with the tuple granularity and in the case of accuracy, we manage the granularity at the level of each measure attribute.

For the following sub-sections, consider the SQL specifications for the multidimensional operations presented in Section 2.1.

4.1 Estimations

In this section we present, for each multidimensional operation, a general formula for estimating the quality value of the result of its application.

Freshness. In this context, it is not worth considering the cost (duration) of the operations because it is depreciable in comparison to the loading time of a DM and to the period of time between two loadings. We consider facts' freshness relevant only, since we assume dimensions have rather stable information.

Estimations for RA Operations:

- Projection, Selection, Aggregation: The original fv is maintained by these operations.
- Join (R,S), Union (R,S):

$$fv(Result) = max(fv(R), fv(S)) . \quad \textbf{(9)}$$

Estimations for Multidimensional Operations:

- Dice, Roll-Up, Projection, Change Base: The result has the same freshness value as the input fact table. Dice, Roll-Up, and Change Base operations involve joins between the fact table and dimension tables. However, the freshness of the dimensions does not affect the resulting freshness.

$$fv(Result) = fv(Cell) . \quad \textbf{(10)}$$

- Drill Across: It involves a join between the fact table of the original cube (Cell) and another fact table (Cell'). The dimensions that are also involved in the join, do not affect the resulting freshness.

$$fv(Result) = max(fv(Cell), fv(Cell')) . \quad \textbf{(11)}$$

- Union: The union relational operator is applied over two sets of tuples from different fact tables. The formula is the same as **(11)**.

Accuracy. We consider as relevant the facts' accuracy and not the dimensions' one because the dimensions' values are rather stable and we assume they have a good accuracy, while the quantity of measures' values is constantly growing and these values are the main basis for the decision making process.

Estimations for RA Operations:

- Projection, Selection, Aggregation: The original av is maintained by these operations.
- Join (R,S):

$$av(Result) = av(R) * av(S) . \quad \textbf{(12)}$$

- Union (R,S): A weighted average of input tables' accuracy values, according to the tuple quantity of each input table, is done.

$$av(Result) = (av(R) * |R| + av(S) * |S|) / |R| + |S| . \quad \textbf{(13)}$$

Note that these estimations are proposed for a relation granularity.

Estimations for Multidimensional Operations: In this case, the join operations never affect the resulting accuracy values due to the granularity we manage, therefore we do not apply the estimation proposed for the RA join. When there is a join between fact tables, each measure attribute maintains its accuracy value.

- Dice, Roll-Up, Projection, Change Base: These operations maintain the accuracy values for each measure attribute.

$$\text{av(Result, Measure}_i\text{) = av(Cell, Measure}_i\text{)} . \qquad \textbf{(14)}$$

- Drill Across: It maintains the accuracy values for each measure attribute.

$$\text{av(Result, Cell.Measure}_i\text{) = av(Cell, Measure}_i\text{)} . \qquad \textbf{(15)}$$
$$\text{av(Result, Cell'.Measure}_j\text{) = av(Cell', Measure}_j\text{)} .$$

- Union: For each measure attribute, we do a weighted average as in (13).

$$\text{av(Result, Measure}_i\text{) = (av(Cell, Measure}_i\text{) * |Cell| +} \qquad \textbf{(16)}$$
$$\text{av(Cell', Measure}_i\text{) * |Cell'|) / |Cell| + |Cell'|} .$$

Availability. Given a multidimensional operation, for availability to be true, we need that all necessary data for answering the corresponding SQL query are available. Therefore, in the estimation of this property the dimensions' information has the same incidence as the facts' information.

- Projection, Roll-Up: These operations have as input only one fact table with one associated availability value.

$$\text{vv(Result) = vv(Cell)} . \qquad \textbf{(17)}$$

- Dice: This operation includes a predicate P over an attribute of a dimension table, which may not be its key. Therefore, for executing this operation we need not only the availability of the fact table, but also the availability of the mentioned dimension table. However, all these tables belong to the same DM, therefore, assuring availability of the fact table is enough. The formula is the same as **(17)**.

- Change Base: For this operation not only the fact table must be available, but also the original and new dimension tables, so that the join between them can be done. The new dimension tables may belong to a different DM.

$$\text{vv(Result) = vv(Cell) AND vv(NewLevel}_1\text{) AND … AND} \qquad \textbf{(18)}$$
$$\text{vv(NewLevel}_n\text{)} .$$

- Drill Across, Union: In these operations the resulting data comes from both input fact tables, which may belong to different DMs. Therefore, the availability value of the result is equal to the "AND" of the availability values of the input fact tables.

$$\text{vv(Result) = vv(Cell) AND vv(Cell')} . \qquad \textbf{(19)}$$

4.2 Calculations

In this section we present, for each multidimensional operation, a general formula for calculating properties values of the result.

For calculations we use a tuple granularity, thus the property values are affected by an operation when it generates one tuple from the combination of two or more tuples. Therefore, the RA operations that can affect calculations are join and aggregation.

We use the ideas of estimations for RA operations existing in the literature for the cases of join and union, but in most cases we propose calculations that are specific for our context and granularity.

Freshness.
Calculation of fv_by_tuple. After the application of a multidimensional operation, we can calculate the fv by tuple.

- Projection, Change Base, Dice: These operations do not affect tuples' freshness values. Dice and Change Base operations involve joins between fact table and dimension tables. However, the freshness of dimension tuples does not affect freshness of the resulting tuples.

  $$\text{fv\_by\_tuple}(t_1) = \text{fv\_by\_tuple}(t_2)\ . \qquad \textbf{(20)}$$

  for all $<t_1,t_2>$, where t_1 is a Result tuple and t_2 is the corresponding Cell tuple.
- Union: The union relational operator is applied over two sets of tuples from different fact tables, however, when managing tuple granularity, the freshness values are not affected.

  $$\text{fv\_by\_tuple}(t_1) = \text{fv\_by\_tuple}(t_2)\ . \qquad \textbf{(21)}$$
  $$\text{fv\_by\_tuple}(t_3) = \text{fv\_by\_tuple}(t_4)\ .$$

  for all $<t_1,t_2>$ where t_1 is a Result tuple and t_2 is the corresponding Cell tuple, for all $<t_3,t_4>$, where t_3 is a Result tuple and t_4 is the corresponding Cell' tuple.
- Roll-Up: This operation generates tuples that are aggregations from input tuples.

  $$\text{fv\_by\_tuple}(t) = \max(\text{fv\_by\_tuple}(u))\ . \qquad \textbf{(22)}$$

  for all t, Result tuple, for all $u \in T$, where T is the tuple set of Cell grouped in t.
- Drill Across: Each resulting tuple is generated by a join that involves two fact tables (Cell and Cell'), therefore its freshness is calculated from the freshness of the corresponding tuples of the input fact tables.

  $$\text{fv\_by\_tuple}(t) = \max(\text{fv\_by\_tuple}(u), \text{fv\_by\_tuple}(u'))\ . \qquad \textbf{(23)}$$

  for all t, tuple of Result, where $u \in$ Cell, and $u' \in$ Cell', are the tuples whose values participate in t.

Calculation of fv. After the application of a multidimensional operation, we can calculate the fv of a relation from the fv_by_tuple of its tuples.

$$\text{fv(Result)} = \max(\text{fv\_by\_tuple}(t)) \,. \qquad \textbf{(24)}$$

for all t, tuple of Result.

Note that in the case of Projection and Change Base the fv of the cube is maintained, while in the case of the other operations it may change.

Accuracy.

Calculation of av_by_tuple. Since the granularity managed for accuracy is of tuple and attribute, the RA join operator does not affect the accuracy of the resulting tuples. The value of each measure attribute of the result comes from only one of the input fact tables, therefore the accuracy of each tuple for each measure attribute is maintained by the join.

- Projection, Change Base, Dice: These operations do not affect the tuples' accuracy values for each measure attribute.

$$\text{av\_by\_tuple}(t_1[\text{Measure}_i]) = \text{av\_by\_tuple}(t_2[\text{Measure}_i]) \,. \qquad \textbf{(25)}$$

  for all $<t_1,t_2>$, where t_1 is a tuple of Result and t_2 is the matching tuple of Cell.

- Union: Analogously to the case of freshness, the union relational operator is applied but it does not affect tuples' accuracy values. The accuracy of each tuple in each measure of the result is the same it was in the corresponding input table.

$$\begin{aligned}\text{av\_by\_tuple}(t_1[\text{Measure}_i]) &= \text{av\_by\_tuple}(t_2[\text{Measure}_i]) \,. \qquad \textbf{(26)}\\ \text{av\_by\_tuple}(t_3[\text{Measure}_j]) &= \text{av\_by\_tuple}(t_4[\text{Measure}_j]) \,.\end{aligned}$$

  for all $<t_1,t_2>$, where t_1 is a tuple of Result and t_2 is the matching tuple of Cell, for all $<t_3,t_4>$, where t_3 is a tuple of Result and t_4 is the matching tuple of Cell'.

- Drill Across: It does not affect tuples' accuracy values for each measure attribute.

$$\begin{aligned}\text{av\_by\_tuple}(t_1[\text{Measure}_i])&=\text{av\_by\_tuple}(t_2[\text{Measure}_i]) \,. \qquad \textbf{(27)}\\ \text{av\_by\_tuple}(t_1[\text{Measure}_j])&=\text{av\_by\_tuple}(t_3[\text{Measure}_j]) \,.\end{aligned}$$

  for all $<t_1,t_2,t_3>$, where t_1 is a tuple of Result, t_2 is the corresponding tuple of Cell, and t_3 is the corresponding tuple of Cell'.

- Roll-Up: This operation generates tuples that are aggregations of input tuples. This calculation is made with the same criteria we use in the calculation of the av of a relation (we explain it below).

$$\begin{gathered}\text{av\_by\_tuple}(t[\text{Measure}_i]) = (\text{av\_by\_tuple}(u_1[\text{Measure}_i]) \,* \qquad \textbf{(28)}\\ \text{digits}(u_1[\text{Measure}_i]) + \ldots + \text{av\_by\_tuple}(u_n[\text{Measure}_i]) \,*\\ \text{digits}(u_n[\text{Measure}_i])) \,/\, \text{digits}(u_1[\text{Measure}_i])+\ldots+\text{digits}(u_n[\text{Measure}_i]) \,.\end{gathered}$$

  for all t, tuple of Result, for all $u_i \in T$, where T is the set of tuples of Cell grouped in t, where $u_i[\text{Measure}_i]$ is the value of the attribute Measure_i in tuple u_i, and digits(n) returns the quantity of digits of number n.

Calculation of av: The av of a relation is calculated from the av_by_tuple of its tuples. For each measure attribute, we make a weighted average, taking into account the values of the measure attribute (multiplying by the number of digits), since we consider that the accuracy of greater values must have more influence on the accuracy of the whole table.

$$\text{av}(\text{Result},\text{Measure}_i) = (\text{av\_by\_tuple}(\text{Cell}, \text{Measure}_i,t_1) * \text{digits}(t_1.\text{Measure}_i) +\ldots+ \text{av\_by\_tuple}(\text{Cell}, \text{Measure}_i,t_n) * \text{digits}(t_n.\text{Measure}_i)) / \text{digits}(t_1.\text{Measure}_i) +\ldots+ \text{digits}(t_n.\text{Measure}_i) . \quad \textbf{(29)}$$

where $t_1 \ldots t_n$ are all the tuples of Result, t_i.Measure$_i$ is the value of the attribute Measure$_i$ in tuple t_i, and digits(n) returns the quantity of digits of number n.
Note that in the case of Projection and Change Base the av of each measure is maintained, while in the cases of the other operations it may change.

Availability. In our approach, the calculation of this property has no sense because the availability values never depend on the tuples obtained in each operation, but only on the DMs that are involved in the query. Therefore, only estimations can be done.

5 Conclusion

In this work we propose a mechanism for adding quality properties meta-information to an OLAP system. We state a very specific scenario, where the system is implemented as ROLAP and each DM consists of a set of cubes that correspond to the same fact. For the definition of this scenario, we base on the work presented in [8], [10]. The quality properties we manage are freshness, accuracy and availability. We propose a set of formulas for estimating and calculating the values of these properties for any possible multidimensional query posed by the user. Estimations and calculations are both useful for the user. Estimations can be used for changing the query so that a better quality is obtained, and calculations provide a more exact meta-information that may be very valuable at the moment of making decisions.

For the performed study, we focused on a given set of SQL queries and on data with certain given characteristics, e.g. the measure attributes. Such preconditions allowed us to obtain interesting results, such as some formulas for accuracy that take into account the values of the attributes. They also showed how some operations in general do not affect the quality values, while other ones have a great incidence.

This study also shows the feasibility of applying techniques of quality evaluation to an OLAP environment, emphasizing on the main characteristics of these systems.

Future work will focus on considering user quality requirements and managing the system quality for maintaining their satisfaction. This problem leads to extend the scope of our environment, such that the sources of the DMs and the data transformations between them be considered. Another aspect that may be addressed is the extension of the present study to other quality properties.

Acknowledgements

Our work has been supported by the Spanish Research Program PRONTIC and FEDER under project TIN2005-05406, and by Comisión Sectorial de Investigación Científica from Universidad de la República, Montevideo, Uruguay.

References

1. Naumann, F.; Leser, U.; Freytag, J.C.: Quality-driven Integration of Heterogenous Information Systems. VLDB 1999: 447-458
2. Marotta, A.; Ruggia, R.: Quality Management in Muti-Source Information Systems. II Workshop de Bases de Datos, Jornadas Chilenas de Computación (JCC'03), Chile. Nov. 03
3. Marotta, A.; Ruggia, R.: Managing Source Quality Changes in Data Integration Systems. Second International Workshop on Data and Information Quality (DIQ'05) (in conjunction with CAISE). June, 14th. 2005, Porto, Portugal
4. Bach Pedersen, T.; Shoshani, A.; Gu, J.; Jensen, C.: Extending OLAP Querying to External Object Databases. Int. Conf. on Information and Knowledge Management. CIKM 2000.
5. Pedersen, D.; Riis, K.; Bach Pedersen, T.: Query optimization for OLAP-XML federations. ACM Fifth International Workshop on Data Warehousing and OLAP, Nov. 2002, USA.
6. Motro, A.; Rakov, I.: Estimating the Quality of Databases. International Conference on Flexible Query Answering Systems. FQAS 1998: 298-307
7. Peralta, V. ; Ruggia, R.; Kedad, Z.; Bouzeghoub, M.: A Framework for Data Quality Evaluation in a Data Integration System. 19º Simposio Brasileiro de Banco de Dados (SBBD'2004). Brasil, October 2004
8. Abelló, A.; Samos, J.; Saltor, F.: Implementing Operations to Navigate Semantic Star Schemas (© ACM). In 6th International Workshop on Data Warehousing and OLAP (DOLAP 2003). New Orleans (USA), November 2003.
9. Abelló, A.; Samos J.; Saltor F.: YAM2 (Yet Another Multidimensional Model): An Extension of UML. International Database Engineering & Applications Symposium, IDEAS'02, July 17-19, 2002, Edmonton, Canada.
10. Romero, O.; Abelló, A.: Improving automatic SQL translation for ROLAP tools. In Proceedings of Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2005). Granada (Spain), September 2005. Pages 123-130. Thomson Editores, ISBN 84-9732-434-X
11. Piedrabuena, F.; Tercia, S.; Vazquez, G.: Federation Ontology for Multi-Source Information Systems. Internal Report. Instituto de Computación, Facultad de Ingeniería, Uruguay. 2005
12. Lee, Y.W.; Strong, D.M.; Kahn, B.K.; Wang, R.Y.: AIMQ: A Methodology for Information Quality Assessment. Information & Management, published by Elsevier Science (North Holland). (Accepted in November 2001)
13. Theodoratos, D.; Bouzeghoub, M.: Data Currency Quality Factors in Data Warehouse Design. In Proc. of the Int. Workshop on Design and Management of Data Warehouses (DMDW'99), Germany, 1999.
14. Bouzeghoub, M.; Peralta, V.: A Framework for Analysis of Data Freshness. 1st Int. Workshop on Information Quality in Information Systems (IQIS). Paris, France, June 2004.
15. Motro, A.: Accommodating Imprecision in Database Systems: Issues and Solutions. ACM SIGMOD Record (Special issue on directions for future DBMS research and development), Vol. 19, No. 4, December 1990, pp. 69--74
16. Strong, D.M.; Lee, Y.W.; Wang, R.Y.: Data Quality in Context. Communications of the ACM. May 1997/Vol.40, No.5
17. Availability. http://availability.com/. Last accessed: Oct. 9, 2005.