Modelling and Optimisation Issues for Multidimensional Databases

Panos Vassiliadis **Spiros Skiadopoulos**

National Technical University of Athens

Department of Electrical and Computer Engineering

Computer Science Division

Knowledge and Database Systems Laboratory

Zografou 15773, Athens, Greece

Tel: +301-772-1402

Fax: +301-772-1442

{pvassil,spiros}@dbnet.ece.ntua.gr

Abstract

On-Line Analytical Processing (OLAP) is a trend in database technology based on the multidimensional view of data. Although multidimensional data cubes form the basic logical data model for OLAP applications, there seems to be no agreement on a common model for cubes. In this paper we propose, a logical model for cubes based on the key observation that a cube is not a self-existing entity, but rather a view over an underlying data set. The model is powerful enough to capture all the commonly encountered OLAP operations such as *selection*, *roll-up* and *drill-down*, through a sound and complete algebra. We accompany our model with results on processing cube operations and provide syntactic characterisations for the problem of cube usability (i.e., the problem of using the tuples of a cube to compute another cube). As part of the solution to this problem, we have developed algorithms to check whether (a) the marginal conditions of two cubes are appropriate for a rewriting, in the presence of aggregation hierarchies and (b) an implication exists between two selection conditions that involve functionally dependent attributes (levels of aggregation in our context). For the latter, we have extended the well-known set of axioms for conjunctive query containment [Ullm89] with axioms describing the role of the functional dependencies. Finally, we present a rewriting algorithm for the cube usability problem.

1. Introduction

On-Line Analytical Processing (OLAP) is a trend in database technology based on the multidimensional view of data. Although multidimensional data cubes form the basic logical data model for OLAP applications, up to now, no common agreement has been obtained on the elements of a cube model. Several industrial standards already exist [OLAP97, TPC98, Meta97, MS98], yet, apart for the last one, none of them seems to propose a well-founded model for OLAP databases. In academia, several proposals on the modelling of cubes also exist [AGS95, LW96, GL97, BPT97, CT97, Lehn98, Vass98]. Despite all these efforts, we feel that several key characteristics of a cube model have not been stressed, neither by the academia nor the industry (see [VaSe99] for a complete discussion). To this end, we present a *logical* model for cubes. This model extends the proposal of [Vass98] in a more formal and systematic way. It deals with all the commonly encountered entities of a multidimensional model (dimension hierarchies, data cubes and cube operations) without being restricted from their physical implementation (e.g., ROLAP or MOLAP architectures). One of our key observations is that *a cube is not a self-existing entity, but rather a view* (materialised or not) *over an underlying data set*. This property allows us to develop complex operations, not dealt by other models so far (e.g., the drill-down operation and the change of aggregation function).

To our knowledge, existing OLAP tools behave in an "extensional" fashion. Cubes are treated simply as sets of tuples, ignoring the fact that they are produced as queries over an underlying detailed data set (e.g., the fact table of a data warehouse). Our framework, instead, suggests a different strategy: we keep the "history" of performed selections and thus, we are able to compute a new cube taking into account its "intentional" description. Therefore, we can define more complex operations (such as drill-down) and sequences of operations, which are not covered by other models. Our model is accompanied by an algebra powerful enough to capture the usual OLAP operations such as (a) *selection* over a cube, (b) *roll-up*, which means aggregation over a cube to coarser granularities of information and (c) *drill-down*, which involves de-aggregation of a specific cube and presentation of more detailed information.

The contribution of this paper lies not only in terms of expressiveness, but also we present results on optimisation issues for multidimensional databases. We investigate the *cube usability problem*, a variant of the relational view usability problem, for multidimensional cubes. We accompany our framework with optimisation techniques for the cube usability problem that enable the exploitation of existing cubes in order to compute new cubes. To handle the cube usability problem, we extend well-known techniques already found in the relational context on the containment of selection conditions [Ullm89]. We have observed that although quite a lot of work has been performed in the field of query containment and view usability in the context of relational databases [DJLS96, GHQ95, NSS98, CKPS95], there exist no results to exploit the information about dimension hierarchies in the context of multidimensional databases. We present results on two major topics. First, we tackle the problem of containment of two selections, taking into account their marginal conditions in the presence of dimension hierarchies. Secondly, we come up with a set of axioms to characterise containment for expressions involving functionally dependent attributes. Although several results already exist to characterise query containment between expressions involving one domain [Ullm89], to our knowledge, no results exist for expressions involving different functionally dependent levels.

This paper is organised as follows. In Section 2 we present the logical cube model. Section 3 presents optimisation issues. Finally, in Section 4 we discuss our results and present future work. All the proofs are found in the Appendix.

2. Cubes for Multidimensional Databases

In this section we present the basic entities and operations of our model. Entities involve dimensions, data sets and cubes. Operations involve selections and change in the granularity of data. This model extents previous proposals of [Vass98, CT97, Lehn98].

One of the main characteristics of OLAP applications is the *multidimensional view of data* in the perception of the user, which considers that information is stored in a *multi-dimensional array*, called *Cube* or *HyperCube*. Thus, a *Cube* is a group of data cells. Each cell is uniquely defined by the corresponding values of the dimensions of the cube (i.e., its coordinates). The contents of the cell are named *measures* and represent the measured values of the real world. Measures are functionally dependent, in the relational sense, on the dimensions of the cube.

A *dimension* is defined in [OLAP97] as "a structural attribute of a cube that is a list of members, all of which are of a similar type in the user's perception of the data". Informally, a dimension models all the possible ways in which the user can group the detailed information stored in the multidimensional database with respect to a specific context. Each dimension has an associated *hierarchy of levels* of aggregated data i.e., it can be viewed from different levels of detail. Formally, a *dimension* D is a lattice $(\mathbf{L},\prec)$: $\mathbf{L}=(L_1,\ldots,L_n,ALL)$. We require that the upper bound of the lattice is always the level ALL, so that we can group all the values of the dimension into the single value 'all'. The lower bound of the lattice is called the *detailed level* of the dimension. For instance, let us consider the dimension Date of Figure 2.5. Levels of dimension Date are Day, Week, Month, Year and ALL. Day is the most detailed level. Level ALL is the most coarse level for all the dimensions. Aggregating to the level ALL of a dimension ignores the respective dimension in the grouping (i.e., practically groups the data with respect to all the other dimensions of the cube, except for this particular one).

The relationship between the values of the dimension levels is achieved through the use of the set of $anc_{L_1}^{L_2}$ functions. A function $anc_{L_1}^{L_2}$ assigns a value of the domain of L_2 to a value of the domain of L_1. For instance $anc_{Month}^{Year}(Feb\text{-}97)=1997$.

The major multi-dimensional operations are *selection* and *navigation*. Selection is used whereby a criterion is evaluated against the data or levels of a dimension in order to restrict the set of retrieved data. Navigation is a term used to describe the processes employed by users to explore a cube interactively by changing the granularity of the multi-dimensionally viewed data [OLAP97]. Possible navigation operations, which can be applied to a cube, are: (a) *Roll-up* which corresponds to the aggregation of data from a lower to a higher level of granularity within a dimension's hierarchy, (b) *Drill-Down* which is the inverse of roll-up and allows the de-aggregation of information moving from higher to lower levels of granularity and (c) *Slice* which corresponds to the grouping of data with respect to a subset of the dimensions of a cube. For instance, let us consider the dimension Date; aggregating from Month to Year is a roll-up operation and de-aggregating from Month to Day is a drill-down operation. In our model, the slice operation is modelled as a roll-up to level ALL.

In our model, we denote sets of tuples under a specific schema by the term *data set*. Moreover, we assume the existence of a *detailed data set*, i.e., a data set that is defined at the finest levels of granularity for all its dimensions. This detailed data set is the central source of data, which will populate any cubes produced during an OLAP session (e.g., a fact table in a data warehouse).

One of our key observations is that a *cube* is not a self-existing entity (as commonly encountered in the literature), but rather a view over an underlying detailed data set. As usual, a view (and thus a cube) can be either materialised or not. Therefore, a cube can be seen either as a data set or simply a query. In our model, we retain this dual nature formally; a cube is not only a set of tuples, but also has a definition. This definition is a query that reduces the computation of the cube to a set of operations over the initial materialised detailed data set.

Formally, a *cube* c over the schema $[L_1,\ldots,L_n,M_1,\ldots,M_m]$, is an expression of the form: $c=(DS^0, \varphi, [L_1,\ldots,L_n,M_1,\ldots,M_m], [agg_1(M_1^0),\ldots,agg_m(M_m^0)])$, where DS^0 is a detailed data set over the schema $S=[L_1^0,\ldots,L_n^0,M_1^0,\ldots,M_k^0]$, $m\leq k$, φ is a detailed selection condition, $M_1^0,\ldots,M_m^0$ are detailed measures, $M_1,\ldots,M_m$ are aggregated measures, L_i^0 and L_i are levels such that $L_i^0 \prec L_i$, $1\leq i\leq n$ and agg_i, $1\leq i\leq m$ are aggregated functions from the set $\{sum,min,max,count\}$.

Intuitively, to compute a cube, first we apply the selection condition to the detailed data set. Then, we replace the values of the levels for the tuples of the result, with their respective ancestor values at the levels of the schema of the cube and group them into a single value for each measure, through the application of the appropriate aggregate function. Note that a detailed data set can be trivially expressed as a cube, having a `true` selection condition and an arbitrary aggregate function. For instance, the cube of the detailed data set DS^0 of Figure 2.1 is expressed as: $c^0=(DS^0, true, [day,day,item,salesman,city,sales], sum(sales))$.

This approach introduces a powerful expression mechanism, able to directly capture operations like drill-down and change of aggregate function and thus, aimed towards the modelling of sequences of operations, as normally encountered in OLAP systems. To our knowledge, no other model can capture these operations directly. The reduction of a cube's definition to a normalised form seems to be the only alternative that directly achieves this kind of functionality.

Formally, the model consists of the following elements:

- Each *dimension* D is a lattice $(\mathbf{L},\prec)$ such that: $\mathbf{L}=(L_1,\ldots,L_n,ALL)$ is a finite subset of *levels* and $\prec$ is a partial order defined among the levels of $\mathbf{L}$, such that $L_1 \prec L_i \prec ALL$ for every $1\leq i\leq n$.
- A family of functions $anc_{L_1}^{L_2}$ satisfying the following conditions (extending [CT97]):
  1. For each pair of levels L_1 and L_2 such that $L_1 \prec L_2$ the function $anc_{L_1}^{L_2}$ maps each element of $dom(L_1)$ to an element of $dom(L_2)$.
  2. Given levels L_1, L_2 and L_3 such that $L_1 \prec L_2 \prec L_3$, the function $anc_{L_1}^{L_3}$ equals to the composition $anc_{L_1}^{L_2} \circ anc_{L_2}^{L_3}$.
  3. For each pair of levels L_1 and L_2 such that $L_1 \prec L_2$ the function $anc_{L_1}^{L_2}$ is monotone, i.e., $\forall x,y \in dom(L_1), L_1 \prec L_2: x<y \Rightarrow anc_{L_1}^{L_2}(x) \leq anc_{L_1}^{L_2}(y)$.

4. For each pair of levels L_1 and L_2 the $anc_{L_1}^{L_2}$ function determines a set of finite equivalence classes X_i such that: $\forall x, y \in dom(L_1), L_1 \prec L_2: anc_{L_1}^{L_2}(x) = anc_{L_1}^{L_2}(y) \Rightarrow x, y$ belongs to the same X_i.
5. The relationship $desc_{L_1}^{L_2}$ is the inverse of the $anc_{L_1}^{L_2}$ function -i.e., $desc_{L_1}^{L_2}(l) = \{x \in dom(L) : anc_{L_1}^{L_2}(x) = l\}$.

– Each *data set* DS over a schema $S=[L_1, \ldots, L_n, M_1, \ldots, M_m]$ is a finite set of tuples over S such that $[L_1, \ldots, L_n]$ comprise a primary key (in the usual sense).

– Each *selection condition* φ is a formula in disjunctive normal form. An *atom* of a selection condition is true, false or an expression of the form $x\ \theta\ y$, where θ is an operator from the set $(>, <, =, \geq, \leq, \neq)$ and each of x and y can be one of the following: (a) a level L, (b) a value l, (c) an expression of the form $anc_{L_1}^{L_2}(L_1)$ where $L_1 \prec L_2$ and (d) an expression of the form $anc_{L_1}^{L_2}(l)$ where $L_1 \prec L_2$ and $l \in dom(L_1)$. The *detailed equivalent of* φ, denoted by φ^0, is a selection condition obtained through the following procedure: for each occurrence of a level name L in φ, we substitute it with the equivalent expression $anc_{L^0}^{L}(L^0)$, where L^0 is the detailed level of the dimension to which L belongs.

– Each *cube* c over the schema $[L_1, \ldots, L_n, M_1, \ldots, M_m]$, is an expression of the form: $c=(DS^0, \varphi, [L_1, \ldots, L_n, M_1, \ldots, M_m], [agg_1(M_1^0), \ldots, agg_m(M_m^0)])$, where DS^0 is a detailed data set over the schema $S=[L_1^0, \ldots, L_n^0, M_1^0, \ldots, M_k^0]$, $m \leq k$, φ is a detailed selection condition, $M_1^0, \ldots, M_m^0$ are detailed measures, $M_1, \ldots, M_m$ are aggregated measures, L_i^0 and L_i are levels such that $L_i^0 \prec L_i$, $1 \leq i \leq n$ and agg_i, $1 \leq i \leq m$ are aggregated functions from the set {sum,min,max,count}. The expression characterising a cube has the following formal semantics:

$$c=\{x \in Tup(L_1, \ldots, L_n, M_1, \ldots, M_m) \mid \exists y \in \varphi(DS^0), x[L_i]=anc_{L_i^0}^{L_i}(y[L_i^0]),\ 1 \leq i \leq n,$$
$$x[M_j]=agg_j(\{q \mid \exists z \in \varphi(DS^0), x[L_i]=anc_{L_i^0}^{L_i}(z[L_i^0]), 1 \leq i \leq n, q=z[M_j^0]\}), 1 \leq j \leq m\}.$$

– The *Cube Algebra* (CA) is composed of three operations (consider a cube $c^a=(DS^0, \varphi^a, [L_1^a, \ldots, L_n^a, M_1^a, \ldots, M_m^a], [agg_1^a(M_1^0), \ldots, agg_m^a(M_m^0)]$ over which the operations are performed):

1. *Navigate*: Let $S=[L_1, \ldots, L_n, M_1, \ldots, M_m]$ be a schema and $agg_1, \ldots, agg_m$ be aggregate functions. If L_i^a and L_i belong to the same dimension D_i and $agg_i \in \{sum, min, max, count\}$ then, navigation is defined as follows:

$$nav(c^a, S, agg_1, \ldots, agg_m) = (DS^0, \varphi^a, S, [agg_1(M_1^0), \ldots, agg_m(M_m^0)]).$$

2. *Selection*: Let φ be a selection condition applicable to c^a. Then, we define the *selection* operation as:

$$\sigma_\varphi(c^a) = (DS^0, \varphi^a \wedge \varphi^0, [L_1^a, \ldots, L_n^a, M_1^a, \ldots, M_m^a], [agg_1^a(M_1^0), \ldots, agg_m^a(M_m^0)])$$

where φ^0 is the detailed equivalent of the selection condition φ.

3. *Split measure*: Let M be a measure of the schema of the cube c. Without loss of generality, let us assume that M is M_m. Then *split measure* is defined as follows:

$$\pi_{M_m}(c^a) = (DS^0, \varphi^a, [L_1^a, \ldots, L_n^a, M_1^a, \ldots, M_{m-1}^a], [agg_1^a(M_1^0), \ldots, agg_{m-1}^a(M_{m-1}^0)]).$$

Day	Title	Salesman	Store	Sales
6-Feb-97	Symposium	Netz	Paris	7
18-Feb-97	Karamazof brothers	Netz	Seattle	5
11-May-97	Ace of Spades	Netz	Los Angeles	20
3-Sep-97	Zarathustra	Netz	Nagasaki	50
3-Sep-97	Report to El Greco	Netz	Nagasaki	30
1-Jul-97	Ace of Spades	Venk	Athens	13
1-Jul-97	Piece of Mind	Venk	Athens	34

Figure 2.1: Detailed Data Set $\mathtt{DS}^0$.

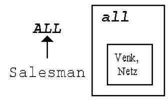


Figure 2.2: Dimension `Person`.

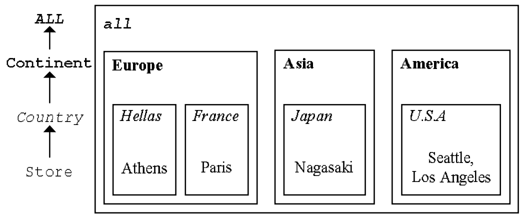


Figure 2.3: Dimension `Location`.

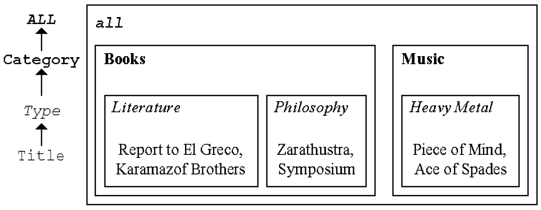


Figure 2.4: Dimension `Product`.

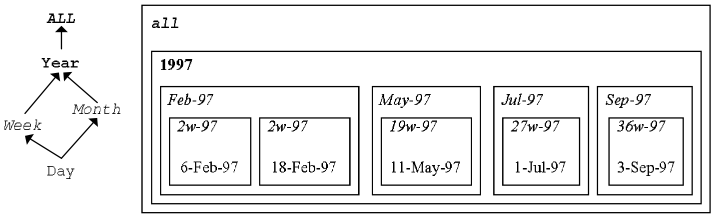


Figure 2.5: Dimension `Date`.

Theorem 2.1. The Cube Algebra `CA` is *sound* (i.e., the result of all the operations is always a cube) and *complete* (i.e., any valid cube can be computed as the combination of a finite set of `CA` operations).

Example 2.1: To motivate the discussion we customise the example presented in [MS98] to an international publishing company with travelling salesmen selling books and CD's to stores all over the world. The database (Figure 2.1) stores information about the sales of a title that a salesman achieved on a particular date and city. The dimensions of our example are `Person` (Figure 2.2), `Location` (Figure 2.3), `Product` (Figure 2.4) and `Date` (Figure 2.5). Measure `Sales` is functionally dependent on dimensions `Date`, `Product`, `Person` and `Location`.

The organisation of information in different levels of aggregation (i.e., dimensions) is in hand because OLAP users are unlikely to directly ask questions about the detailed data that are stored in the database. Instead, they are more interested in aggregated information according to the categorisation groupings.

Following, we present three queries and the respective algebraic representation that could have been a typical sequence of operations during an OLAP session. These queries result in the data of Figure 2.6, Figure 2.7 and Figure 2.8 respectively.

Query 1. Find the maximum sales by month, category of item, salesman and country.

$$\mathtt{c}^1\mathtt{=nav(DS}^0\mathtt{,[Month,Category,Salesman,Country,Max\_val],max(sales))=}$$
$$\mathtt{(DS}^0\mathtt{,true,[Month,Category,Salesman,Country,Max\_val],max(sales)).}$$

Query 2. Find the maximum sales outside the American continent by month, category of item, salesman and country.

$$\mathtt{c}^2\mathtt{=}\sigma_{\mathtt{anc}_{\mathtt{country}}^{\mathtt{continent}}\mathtt{(country)}\neq\mathtt{'America'}}\mathtt{(c}^1\mathtt{)=(DS}^0\mathtt{,anc}_{\mathtt{city}}^{\mathtt{continent}}\mathtt{(City)}\neq\mathtt{'America',}$$
$$\mathtt{[Month,Category,Salesman,Country,Max\_val],max(sales)).}$$

Query 3. Find the summary of sales outside the continent of America by month, type of title and country of store.

$$c^3 = \texttt{nav}(c^2, [\texttt{Month},\texttt{Type},\texttt{All},\texttt{Country},\texttt{Sum\_val}], \texttt{sum(Sales)}) = (DS^0, \texttt{anc}_{\texttt{city}}^{\texttt{continent}}(\texttt{City}) \neq \texttt{'America'}, [\texttt{Month},\texttt{Type},\texttt{All},\texttt{Country},\texttt{Sum\_val}], \texttt{sum(sales)}).$$

Month	Category	Salesman	Country	Max_val
Feb 97	Books	Netz	France	7
Feb 97	Books	Netz	USA	5
May 97	Music	Netz	USA	20
Sept 97	Books	Netz	Japan	50
July 97	Music	Venk	Greece	34

Figure 2.6: Cube c^1 - navigation as roll-up

Month	Category	Salesman	Country	Max_val
Feb 97	Books	Netz	France	7
Sept 97	Books	Netz	Japan	50
July 97	Music	Venk	Greece	34

Figure 2.7: Cube c^2 – Selection

Day	Type	ALL	Country	Sum_val
Feb 97	Philosophy	All	France	7
Sep 97	Philosophy	All	Japan	50
Sep 97	Literature	All	Japan	30
Jul 97	Heavy Metal	All	Greece	47

Figure 2.8: Cube c^3 – Complex sequence of operations

During this particular OLAP session the user has performed:

1. a roll-up from the detailed data set.
2. a selection.
3. a slicing (of dimension `Person`) combined with a drill down (from `Category` to `Type` level) and a change in the aggregation function (from `max` to `sum`).

In the first operation, one can notice that the semantics of the navigation operation allow us to use an arbitrary name (e.g., `Max_val`) for the measure that computes the maximum value per group of aggregation.

In the second operation, notice that the expression $\texttt{anc}_{\texttt{country}}^{\texttt{continent}}(\texttt{Country})$ which is directly applicable to the schema (and data) of the cube c^1 is transformed to its equivalent $\texttt{anc}_{\texttt{city}}^{\texttt{continent}}(\texttt{City})$, that directly applies to the detailed data set DS^0, through the use of the definition of the detailed selection condition.

The presented model stresses the fact that *a cube we can treated both as a query and as a set of tuples*. We believe that this aspect of OLAP was neglected in the previous approaches. In this example, the contribution of treating cubes as views over the detailed data set is eminent. Actually, the fact that we have retained the history of selections permits us to be able to drill-down and change the aggregation function. Otherwise, to perform the drill-down operations we should employ a join operation of c^2 with DS^0. The same also holds for the change in the aggregation function. Using the history of selections we can (a) avoid to perform a costly join operation and (b) possibly further optimise the execution of the operation through the use of already computed cubes. The second possibility will be investigated in Section 3.

As we have already stressed, this is a *logical* model for cubes. We do not advocate that the *physical* computation of the results of an operation should actually be computed all the way back from the detailed data set. Actually, although drill-down and change of aggregation function can be performed directly, only through the use of the semantics of our model, can the selection and roll-up operations be performed over the original cube, without referring to the detailed data set. In the case of selection, it suffices to simply pass all the tuples of the cube from the filter of the applied selection condition. In the case of roll-up to coarser levels of granularity, it also suffices to group the tuples of the cube and apply the appropriate aggregate function. These simple optimisation strategies are generalised in Section 3 with a more powerful approach, capable of detecting whether any cube can be computed from the data of another cube, simply by comparing their definitions.

3. The Cube Usability Problem

Problem description. There are several cases where there is the need to decide whether a view can be recomputed from another view. To name two prominent examples, (a) the OLAP users perform interactive navigations over their data and (b) the data warehouse designer has to choose, among many candidates, which views to materialise. In the first case, the problem is as follows: the OLAP user selects some data and performs an operation over them. The result of the new query can be computed, of course, from the detailed data. Nevertheless, it is possible that previously computed and cached results, or existing materialised views, could also allow the computation of the requested information. In the second case, the designer of the data warehouse needs algorithmic aid in order to decide if he/she is going to materialise any extra (possibly redundant) views in the data warehouse, so that user queries are answered more efficiently. Sometimes, the redundancy can aid in the refreshment of the data warehouse data, too [ThLS99, LSTV99, Gupt97]. Part of the design algorithm, then, is a method that determines whether a view can be used to compute a query (or another view). As a general statement, one could say that the problem lies in whether the computation of a new cube can be performed from an intermediate level of aggregation, than from the detailed data set.

Formally, let $\mathtt{DS}^0$ be a detailed data set. Let also $\mathtt{c}^{\mathtt{old}}$ and $\mathtt{c}^{\mathtt{new}}$ be two cubes defined over $\mathtt{DS}^0$. By definition, cubes $\mathtt{c}^{\mathtt{old}}$ and $\mathtt{c}^{\mathtt{new}}$ can be calculated from $\mathtt{DS}^0$. The *cube usability problem* lies on determining whether the tuples of $\mathtt{c}^{\mathtt{old}}$ can be used to compute cube $\mathtt{c}^{\mathtt{new}}$. It is clear that the cube usability problem is a variant of the *view subsumption* problem, already investigated in the field of relational databases [Ullm97].

Shortcomings of current approaches. Too much effort has been spent, in the past, to tackle the problem of view subsumption and query rewriting in the presence of views [NSS98, CoNS99, DJLS96, CKP95, GHQ95, LMSS95, ChSh96, LaYa85]. Nevertheless, the previous approaches are relational-oriented and lack to deal with specific characteristics of the multidimensional modelling. We will use two examples to demonstrate these shortcomings.

Example 3.1. Intuitively, someone would expect, that in order to solve the cube usability problem, the new cube $\mathtt{c}^{\mathtt{new}}$ should:

1. be defined over the same dimensions with $\mathtt{c}^{\mathtt{old}}$ and at a higher or equal level;

2. be defined over the same measure of DS^0. Moreover, the aggregation functions agg^{new} and agg^{old} should be the same;
3. have a more restrictive selection condition than c^{old}, i.e., φ^{new} is contained in φ^{old} in the usual relational sense.

Checking conditions 1 and 2 is an easy task. To perform the comparison of Condition 3, we need to transform the selection conditions of the two cubes in order to treat them as conjunctive queries [Ullm89]. One could argue that existing relational techniques are adequate to handle this problem. Unfortunately, as we will show, there are cases where those techniques are not sufficient.

Cube c^1

Month	Sales
Feb	5
May	20
Sep	80

Cube c^2

Month	Sales
Feb	**12**
May	20
Sep	80

Cube c^3

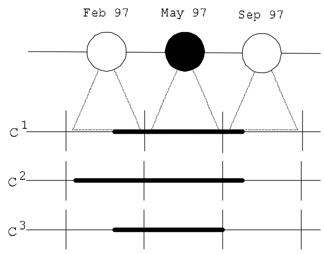
Month	Sales
Feb	5
May	20

φ_1=18-Feb-97≤day≤3-Sep-97 ∧Salesman=Netz

φ_2=6-Feb-97≤day≤3-Sep-97∧Salesman=Netz

φ_3=18-Feb-97≤day≤31-May-97∧Salesman=Netz

(a)



(b)

Figure 3.1: Cube usability problems

Let us consider the detailed data set DS^0 of Figure 2.1. Let c^i, $1\le i\le 3$ be cubes defined as c^i=[DS^0, φ_i, [Month,ALL,ALL,ALL,ALL,ALL,ALL,Sales],sum(sales)]. Figure 3.1a presents the Month level, the Sales measure and the selection conditions for each of the cubes. The problem is whether a new cube c^3 can be computed using the tuples of one of the existing cubes c^1 and c^2. Since Conditions 1, 2 and 3 hold, one could argue that this is feasible. Yet, as we can see in Figure 3.1a, only c^1 can be used to compute c^3. The intuitive explanation of the problem is depicted in Figure 3.1b. There are three horizontal axes defined at the day level, each for one of the cubes c^1, c^2 and c^3. Each bold line denotes the set of days participating in the computation of the respective cube. Cube c^3 is defined at the month level; consequently, we partition the three axes with respect to the function anc_{day}^{month}. As we can see, we have three partitions: Feb'97, May'97 and Sep'97. Cube c^3 can be computed from c^1 because for all the partitions of c^3 (i.e., Feb'97, May'97), cubes c^1 and c^3 cover exactly the same days. This does not hold for c^1 and c^2.

Example 3.2. Suppose the case, where a cube c^1 has a selection condition φ_1=arr.year<dep.year (where arr denotes dimension arrival date and dep denotes the dimension departure date). Suppose also that a cube c^2 is defined at the month level and has a selection condition φ_2=arr.month<dep.month. We can see that cube c^1 can be computed from c^2. This means that if c^2 is materialised we can use its tuples to compute c^1. We are able to perform this kind of reasoning because we take advantage of the relationship

between months and years, expressed through the dimension hierarchies, and the family of `anc` functions. To our knowledge, there is no effort in the view subsumption literature that uses this kind of knowledge.

In Figure 3.2, we depict the problem graphically. As one can see, we represent the tuples of the detailed cube as cells in a 2-dimensional space. The horizontal axis represents the dimension `departure date` and the vertical axis represents the dimension `arrival date` (we focus only on these dimensions taking part in the selection conditions of the example). As one can notice, the set of tuples of the detailed data set, fulfilling the condition `arr.month<dep.month` is a strict superset of the set of tuples fulfilling condition `arr.month<dep.month`.

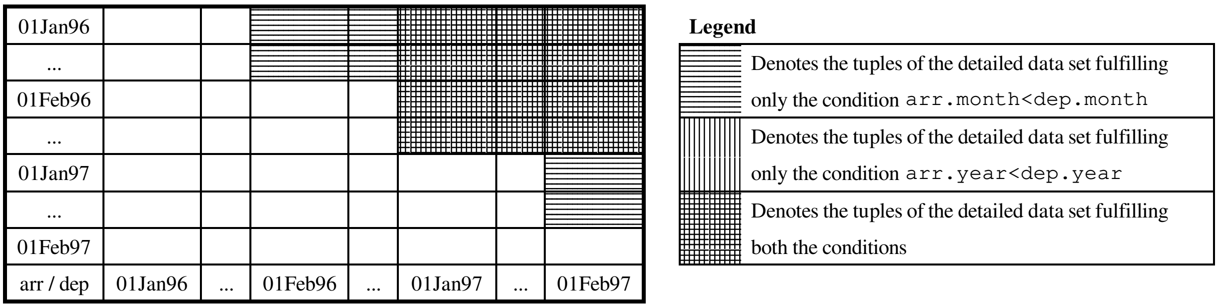


Figure 3.2 Overlapping of selection conditions involving different levels

Contribution In this section, we will show that the cube usability problem is reduced to simple tests and operations. Different tests apply for different classes of queries. We explore the selection conditions of two categories: (a) selection conditions with atoms involving values (i.e., of the form $L\theta l$, $L\theta anc_{L_1}^{L_2}(l)$, etc.) and (b) selection conditions with atoms involving only levels (i.e., of the form $L_1\theta L_2$, $L\theta anc_{L_1}^{L_2}(L_1)$, etc.). We will examine the optimisation issues for the former in Section 3.1 and for the latter in Section 3.2. Finally, Section 3.3 presents a theorem with sufficient criteria and the corresponding rewriting algorithm for both cases of the cube usability problem under consideration.

In the rest of the paper, for reasons of simplicity, we will deal with cubes having only one measure. All our results can be easily extended to cubes having an arbitrary number of measures [NSS98]. Let c^{new}= $(DS^0, \varphi^{new}, [\mathbf{L}^{new}, M^{new}], agg^{new}(M))$ be the new cube and c^{old}=$(DS^0, \varphi^{old}, [\mathbf{L}^{old}, M^{old}], agg^{old}(M))$ be the candidate cube, where $\mathbf{L}^{new}$ and $\mathbf{L}^{old}$ are sets of levels coming from dimension sets $\mathbf{D}^{new}$ and $\mathbf{D}^{old}$ respectively, M^{new} and M^{old} are measures, and finally, agg^{new} and agg^{old} are aggregate functions.

3.1 Equivalent Transformations for Atoms Involving Values

Suppose two levels L^{old} and L^{new}, such that $L^{old} \prec L^{new}$. Function $anc_{L^{old}}^{L^{new}}$ defines a partition over the values of L^{old} with respect to the values of L^{new} (e.g., the partition of `year` to `month`). Suppose now, two atoms a_1 and a_2 over L^{old}, as in the case of Figure 3.1. To perform an aggregation to L^{new}, the two atoms must hold the same ranges of values for each and every partition that L^{new} defines over L^{old}. Generalising this observation, in the case where two selection conditions involve a larger conjunction of atoms, we must:

(a) transform the selection conditions to concrete ranges for each dimension;

(b) reduce the atoms to the same level, using appropriate transformations (so that they can be compared);

(c) check whether the broader selection condition is defined identically for the marginal constraints of the other selection condition.

The following auxiliary definition introduces the notion of *dimension interval*, which is a concrete range over the domain of a certain dimension level.

Definition 3.1: A *dimension interval* (DI) is one of the following (a) `true`, (b) `false` and (c) an expression of the form $l_1 \leq L \leq l_2$, where L is a variable ranging over the level of a dimension and l_1 and l_2 are values. ■

Atom	Dimension Interval	Atom	Dimension Interval
true	true	$anc_L^{L'}(L)<l$	$-\infty<L\leq max(desc_L^{L'}(prev(l)))$
false	false	$l\leq anc_L^{L'}(L)$	$min(desc_L^{L'}(l))\leq L<+\infty$
$anc_L^{L'}(L)=l$	$min(desc_L^{L'}(l))\leq L\leq max(desc_L^{L'}(l))$	$l<anc_L^{L'}(L)$	$min(desc_L^{L'}(next(l)))\leq L<+\infty$
$anc_L^{L'}(L)\leq l$	$-\infty<L\leq max(desc_L^{L'}(l))$		

Figure 3.3: Transformation from atoms to dimension intervals

Figure 3.3 shows how single atoms can be transformed to DI's. Values $-\infty$ and $+\infty$ have the obvious semantics. Moreover, functions `prev` and `next` result in the previous and the following value of l in the domain of L respectively.

In general, to determine whether a cube c^{old} can be used for the computation of c^{new}, we need to partition the detailed level of each dimension according to the respective level of c^{new}. If for each partition of c^{new}, there exists an identical partition of c^{old}, then c^{old} can be used to compute c^{new}. We formalise this relationship between two cubes, through Definition 3.2.

Definition 3.2. $\mathbf{L}$-containment: Let $\mathbf{D}$ be a set of dimensions and φ^{old}, φ^{new} be two selection conditions involving levels only from $\mathbf{D}$. Let $\mathbf{L}$ be a set of levels, each belonging to a different dimension of $\mathbf{D}$. Let also the two cubes $c^{new}=(DS^0,\varphi^{new},[\mathbf{L},M],agg(M))$ and $c^{old}=(DS^0,\varphi^{old},[\mathbf{L},M],agg(M))$, defined over an arbitrary detailed data set DS^0. Selection condition φ^{new} is $\mathbf{L}$-*contained* in φ^{old} (denoted by $\varphi^{new}\subseteq_{\mathbf{L}}\varphi^{old}$) if $c^{new}\subseteq c^{old}$ for any data set DS^0. ■

To tackle the problem of cube usability between cubes of different aggregation granularities, we start by checking the containment of conjunctions of atoms that involve values. Notice that our analysis does not include ≠. This case will be handled in Section 3.3.

Algorithm `Check_Atoms_Usability` of Figure 3.4 takes as inputs two conjunctions of atoms, $\mathbf{a}$ and $\mathbf{b}$, involving only values. It yields `true` if $\mathbf{a}$ $\mathbf{L}$-contains $\mathbf{b}$ with respect to a specific set of levels $\mathbf{L'}$, and `false` otherwise. The algorithm proceeds as follows. Initially, Algorithm `Check_Atoms_Usability` rewrites all atoms of $\mathbf{a}$ and $\mathbf{b}$ to DI's using the transformations of Figure 3.3 (Line 1), reducing all the atoms to DI's in the detailed form. Then, it groups all dimension intervals of $\mathbf{a}$ and $\mathbf{b}$ by dimension level and produces, for every set, a single DI having the most restrictive boundaries. In other words, by now, at most one DI exists per dimension for $\mathbf{a}$ and $\mathbf{b}$. The result is stored in the sets of DI's $\mathbf{a'}$ and $\mathbf{b'}$ respectively (Line 2). Lines 3-4 perform a quick filtering to avoid unnecessary checks. Lines 5-8 check whether there exists a dimension level $D_i.L$ of a in $\mathbf{a'}$ that does not exist in any DI of $\mathbf{b'}$. In that case, the algorithm introduces the dimension interval $-\infty \leq D_i.L^0 \leq +\infty$ to **b′**. Line 9 performs an extra quick check, quickly exiting if **b′** is more restrictive than **a′** (remember that by now the two conjunctions should exactly the same number of DI's).

Finally, Lines 10-19 check if for every DI `b` of **b′** there exists a DI `a` in **a′** such that $\beta \subseteq_{L'} \alpha$ for a certain level $L' \in$ **L′**. Lines 12-13 check whether DI `b` is broader than `a`. Then, the algorithm checks whether the marginal partitions are identical (Lines 14-17). If all these conditions are fulfilled, then the algorithm returns `true`.

For Example 3.1 using Algorithm `Check_Atoms_Usability` we can deduce that φ_1 **L**-contains φ_3 (with respect to level `Month`), while φ_2 does not. Moreover, it is interesting to see that if one considers the `year` level, neither φ_1 nor φ_2 **L**-contains φ_3.

Algorithm `Check_Atoms_Usability`.

Input: Two conjunctions of atoms **a** and **b** involving only values, and a set of levels **L′**.

Output: true if $\mathbf{a} \subseteq_{L'} \mathbf{b}$, false otherwise.

1. Write all atoms of **a** and **b** as DI's using the transformations of Fig. 4 (where the level `L` is the detailed level of each dimension).
2. Group all DI's of **a** and **b** by dimension level and produce for every set a single DI' having the most restrictive boundaries. Let **a′** and **b′** be the result, respectively.
3. If there exists a DI `false` in **a′** Then Return `true`.
4. If there exists a DI `false` in **b′** Then Return `false`.
5. For every DI `a` of **a′**
6.   If `a` is defined over dimension level $D_i.L^0$ that does not exist in any DI of **b′** Then
7.   Introduce DI $-\infty \leq D_i.L^0 \leq \infty$ to **b′**.
8. EndFor
9. If **b′** has more DI's than **a′** Then Return `false`.
10. For every DI $a=(A_s, A_e)$ of **a′**
11.   Let the DI $b=(B_s, B_e)$ of **b′** involving the same dimension with `a`. Let also `L′` be the respective level in **L′**.
12.   Case $A_s < B_s$ or $B_e < A_e$ or `b=false`
13.   Return `false`
14.   Case $A_s \neq \mathrm{min}(\mathrm{desc}^{L'}_{L^0}(\mathrm{anc}^{L'}_{L^0}(A_s)))$ and $A_s \neq B_s$
15.   Return `false`
16.   Case $A_e \neq \mathrm{max}(\mathrm{desc}^{L'}_{L^0}(\mathrm{anc}^{L'}_{L^0}(A_e)))$ and $A_e \neq B_e$
17.   Return `false`
18. EndFor
19. Return `true`

Figure 3.4: Algorithm `Check_Atoms_Usability`

3.2 Equivalent Transformations for Atoms Involving only Levels

Following [Ullm89], we assume the existence of two infinite, totally ordered domains, **L** and **L′** isomorphic to the integers. Let also `f` be a total, monotone function over **L**, mapping the values of domain **L** to the values of domain **L′**. The family of `anc` functions fulfils these requirements.

We assume that we are given a collection of inequalities of the form $X<Y$, $X\le Y$, $X\neq Y$, $f(X)<f(Y)$, $f(X)\le f(Y)$, $f(X)\neq f(Y)$ and equalities of the form $f(X)=f(Y)$. We do not allow equalities of the form $X=Y$. If such a subgoal is found in a query, we substitute every occurrence of X with Y. We also eliminate any pair of inequalities $f(X)\le f(Y)$ and $f(Y)\le f(X)$, where X,Y are distinct variables, with $f(X)=f(Y)$.

We will use the following set of axioms for these inequalities:

A1	$X\le X$	A8	$X\le Z$, $Z\le Y$, $X\le W$, $W\le Y$ and $W\neq Z$ imply $X\neq Y$
A2	$X<Y$ implies $X\le Y$	A9	$X\le Y$ implies $f(X)\le f(Y)$
A3	$X<Y$ implies $X\neq Y$	A10	$f(X)<f(Y)$ implies $X<Y$
A4	$X\le Y$ and $X\neq Y$ imply $X<Y$	A11	$f(X)\neq f(Y)$ implies $X\neq Y$
A5	$X\neq Y$ implies $Y\neq X$	A12	$f(X)\le f(Y)$ and $f(Y)\le f(X)$ implies $f(X)=f(Y)$
A6	$X<Y$ and $Y<Z$ imply $X<Z$	A13	$f(X)=f(Y)$ and $f(Y)\le f(Z)$ implies $f(X)\le f(Z)$
A7	$X\le Y$ and $Y\le Z$ imply $X\le Z$	A14	$f(X)=f(Y)$ and $f(Y)\neq f(Z)$ implies $f(X)\neq f(Z)$
		A15	$f(X)=f(Y)$ implies $f(X)\le f(Y)$

Figure 3.5: Axioms for L-containment checking.

We assume that our models are assignments of integers to variables. Expressions of the form $f(X)$ are also treated as variables. For variables of the form X we apply axioms A1 to A9 and for variables of the form $f(X)$ we apply axioms A1 to A15.

Theorem 3.1: The axioms A1-A15 are sound and complete.

In order to check whether one set of inequalities T follows from another set of inequalities S we compute the closure S^+ by applying the axioms A1-A15 until they no longer generate any new inequalities. Then, we check whether T is a subset of S^+.

3.3 Testing Cube Usability

In this section, we combine the results of Sections 3.1 and 3.2 to provide a test for several cases of cube usability. One can transform any kind of formula using logical transformations [Ende72] to an equivalent formula consisting of disjunctions of conjunctions which do not involve $\neq$ and $\neg$. Theorem 3.2 provides sufficient criteria for a cube c^{old} to be used for the computation of another cube c^{new}. Algorithm `Cube_Usability` describes the specific steps to be followed for this computation.

Theorem 3.2: Suppose a detailed data set $DS^0=[L_1^0,\dots,L_n^0,M^0]$ and two cubes $c^{old}=(DS^0,\varphi_{old},[L_1^{old},\dots,L_n^{old},M_{old}],agg_{old}(M^0))$ and $c^{new}=(DS^0,\varphi_{new},[L_1^{new},\dots,L_n^{new},M_{new}],agg_{new}(M^0))$. If

(a) $agg_{old}=agg_{new}$,

(b) $L_i^{old}\prec L_i^{new}$, $1\le i\le n$, and

(c) one of the following two cases holds for φ_{old} and φ_{new}:

- φ_{old} and φ_{new} involve conjunctions of atoms only of the form $L_i\theta L_j$, all the levels L_i,L_j are higher from the respective levels of the schema of c^{old} (i.e. $L_{i,j}^{old}\prec L_{i,j}$) and φ_{old} belongs to the closure of φ_{new}, or,
- φ_{old} and φ_{new} involve conjunctions of atoms of the form $L\theta l$ and $\varphi_{new}\subseteq_{[L_1^{new},\dots,L_n^{new}]}\varphi_{old}$,

then Algorithm `Cube_Usability` correctly computes c^{new} from the tuples of c^{old}.

Algorithm `Cube_Usability`.

Input: A detailed data set $DS^0=[L_1^0,\ldots,L_n^0,M^0]$ and two cubes $c^{old}=(DS^0,\varphi_{old},[L_1^{old},\ldots,L_n^{old},M_{old}],agg_{old}(M^0))$ and $c^{new}=(DS^0,\varphi_{new},[L_1^{new},\ldots,L_n^{new},M_{new}],agg_{new}(M^0))$ such that φ_{old} and φ_{new} involve either (a) conjunctions of atoms of the form $L\theta l$ or (b) conjunctions of atoms of the form $L\theta L'$ where L and L' are levels and l is a value.

Output: A rewriting that calculates cube c^{new} from the tuples of c^{old}.

1. <u>If</u> all atoms of φ_{old} and φ_{new} involve conjunctions of atoms of the form $L\theta l$ <u>Then</u>
2.   <u>For</u> every atom $a=anc_{L^0}^{L}(L^0)\theta l$ in φ_{new} (or equivalent to this form)
3.   <u>If</u> L^{old} is the respective level in the schema of c^{old} and $L^{old}\prec L$ <u>Then</u>
4.   Transform a to $anc_{L^{old}}^{L}(L^{old})\theta l$
5.   <u>EndIf</u>
6.   <u>ElseIf</u> L^{old} is the respective level in the schema of c^{old} and $L\prec L^{old}$ <u>Then</u>
7.   Transform a to $L^{old}\theta' anc_{L}^{L^{old}}(l)$ where $\theta'=\theta$ except for two cases:
   (a) $a=anc_{L^0}^{L}(L^0)<l$ and $l\neq min(desc_{L}^{L^{old}}(anc_{L}^{L^{old}}(l)))$ where $\theta'=\leq$,
   (b) $a=anc_{L^0}^{L}(L^0)>l$ and $l\neq max(desc_{L}^{L^{old}}(anc_{L}^{L^{old}}(l)))$ where $\theta'=\geq$
8.   <u>EndIf</u>
9.   <u>EndFor</u>
10. <u>EndIf</u>
11.   <u>If</u> all atoms of φ_{old} and φ_{new} involve conjunctions of atoms of the form $a=anc_{L^0}^{L}(L^0)\theta anc_{L^{0\prime}}^{L'}(L^{0\prime})$ (or equivalent to this form), where both L and L' are higher than the respective levels of c^{old} <u>Then</u>
12.   <u>For</u> every atom $a=anc_{L^0}^{L}(L^0)\theta anc_{L^{0\prime}}^{L'}(L^{0\prime})$ in φ_{new}
15.   Transform a to $anc_{L^{old}}^{L}(L^{old})\theta anc_{L^{old\prime}}^{L'}(L^{old\prime})$
16.   <u>EndFor</u>
17. <u>EndIf</u>
18. Apply the transformed selection condition to c^{old} and derive a new data set DS^1.
19. Replace all the values of DS^1 with their ancestor values at the levels of c^{new}, resulting in a new data set DS^2.
20. Aggregate ("group by" in the relational semantics) on the tuples of DS^2, so that we produce c^{new}.

Figure 3.6: Algorithm `Cube_Usability`

Theorem 3.2 tests for usability, pairs of cubes involving conjunctive selection conditions which do not involve $\neq$ and $\neg$. Cubes involving disjunctive selection conditions can be treated in the usual way [Ullm89].

Note also, that the inverse ('and only if') of the theorem does not hold. Suppose the case of a particular dimension `D`, involving two levels `low` and `high`, where the `desc` relationship is a function (meaning that anc_{low}^{high} has an inverse function and the mapping from detailed to ancestor values is `1:1`). Then, although condition (b) of Theorem 3.2 is violated, a cube at the level `high` can be used to compute a cube at the level `low`. Moreover, it is easy to construct an example which shows that the above techniques cannot be applied to a class of queries containing both atoms involving only levels (i.e., $L_1\theta L_2$) and atoms involving levels and values (i.e., $L\theta l$).

Example 3.3. Let c^{new} and c^{old} be the cubes over DS^0 of Figure 2.1 defined as follows.

$$c^{old}=(DS^0,\varphi_{old},[\texttt{Month},\texttt{Country},\texttt{Type},\texttt{Salesman},\texttt{Sum\_old}],\texttt{sum(Sales)})\text{ and}$$

$$c^{new}=(DS^0,\varphi_{new},[\texttt{Month},\texttt{Country},\texttt{Category},\texttt{Salesman},\texttt{Sum\_new}],\texttt{sum(Sales)})$$

where $\varphi_{old}=$`18-Feb-97`$\leq$`day` $\wedge$ `day`$\leq$`3-Sep-97` $\wedge$ $anc_{Item}^{Category}(Item)=$"Books" and $\varphi_{new}=$`1-Mar-97`$\leq$`day` $\wedge$ `day`$\leq$`3-Sep-97` $\wedge$ "Literature"$\leq anc_{Item}^{Type}(Item)$ $\wedge$ $anc_{Item}^{Type}(Item)\leq$"Philosophy".

To check whether c^{new} can be computed from c^{old} we apply Theorem 3.2. The schemata and aggregation functions of the two cubes are compatible (conditions (a), (b) of Theorem 3.2). Moreover, φ_{new} is **L**-contained from φ_{old} with respect to the levels of c^{new}. Following Lines 2-10 of Algorithm `Cube_Usability`, we transform φ_{new} so that it can be applied to the schema of cube c^{old}. The transformations of lines 3-8 result in

$\varphi_{n@o}=$`Mar-97`$\leq$`Month` $\wedge$ `Month`$\leq$`Sep-97` $\wedge$ "Literature"$\leq$`Type` $\wedge$ `Type`$\leq$"Philosophy".

We apply the transformed selection condition to c^{old} (depicted in Figure 3.7a) and derive a new data set DS^1 (depicted in Figure 3.7b). Then, we replace all the values of DS^1 with their ancestor values at the levels of c^{new} (Line 19), resulting in a new data set DS^2 (depicted in Figure 3.7c). Finally, we aggregate the tuples of DS^2 and we produce c^{new} (depicted in Figure 3.7d).

Month	Type	Salesman	Country	Sum_old
Feb-97	Literature	Netz	USA	5
Sep-97	Philosophy	Netz	Japan	50
Sep-97	Literature	Netz	Japan	30

(a)

Month	Type	Salesman	Country	Sum_1
Sep-97	Philosophy	Netz	Japan	50
Sep-97	Literature	Netz	Japan	30

(b)

Month	Category	Salesman	Country	Sum_2
Sep-97	Book	Netz	Japan	50
Sep-97	Book	Netz	Japan	30

(c)

Month	Category	Salesman	Country	Sum_new
Sep-97	Book	Netz	Japan	80

(d)

Figure 3.7: Calculating c^{new} from c^{old}.

4. Discussion and Future Work

We have presented a *logical* model for cubes based on the key observation that a cube is not a self-existing entity, but rather a view over an underlying data set. The proposed model is powerful enough to capture all the commonly encountered OLAP operations such as selection, roll-up and drill-down, through a sound and complete algebra. We have showed how this model can be used as the basis for processing cube operations and have provided syntactic characterisations for the problems of cube usability. Theorem 3.2, which provides these syntactic characterisations, is very important for the usual operations of the model. Two of the most eminent cases are: (a) navigation from a certain cube c to a cube having all its levels higher (or equal) than the respective levels of c and (b) selection over a certain cube c where all the levels acting as variables are higher (or equal) than the levels of c.

Of course, the applicability of Theorem 3.2 is not restricted in these two simple cases. Normally, an OLAP screen contains more than one cubes [MS98]. Thus, an interactive OLAP session produces many cubes which possibly overlap. Computing a new set of cubes can possibly be achieved by using already computed and cached cubes (provided that they fulfil the criteria of Theorem 3.2). Consequently, the results on the problem of cube usability can be used both for the query optimisation and the caching processes. The cube usability results can also be applied in the problem of data warehouse / data mart design [ThLS99], where the optimal set of views (with respect to query and maintenance cost) has to be derived. Testing for cube usability can avoid redundancy in the final data warehouse schema and improve the run-time of the design algorithm [LSTV99].

As future work, we plan to incorporate our results in a system under construction in NTUA. The modelling parts could be extended to take into account aspects of the hierarchy structure (partial ancestor functions, hierarchies that are not well captured as lattices [LS97], etc.). The theoretical results over query processing can be extended to handle optimisation issues for a broader set of selection conditions, partial rewritings and optimisation of the physical execution for cube operations. Finally, a challenging issue is how to devise smarter algorithms for the cube usability problems.

References

[AGS95] R. Agrawal, A. Gupta, and S. Sarawagi, "Modelling multidimensional databases", Technical report, IBM Almaden Research Center, San Jose, California, 1995.

[BPT97] E. Baralis, S. Paraboschi, E. Teniente. Materialized View Selection in a Multidimensional Database. In Proc. of *23rd International Conference on Very Large Data Bases (VLDB)*, Athens, August 1997.

[ChSh96] S. Chaudhuri, K. Shim: Optimizing Queries with Aggregate Views. In Proc. of EDBT 1996. *Proceedings of the 5th International Conference on Extending Database Technology (EDBT-96),* Avignon, France, March 25-29, 1996.

[CKPS95] S. Chaudhuri, S. Krishnamurthy, S. Potamianos, and K. Shim, "Optimizing queries with materialized views", *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, IEEE Computer Society, pp. 190-200, Taipei, March 1995.

[CoNS99] S. Cohen, W. Nutt, A. Serebrenik: Rewriting Aggregate Queries Using Views. *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Philadelphia, Pennsylvania. ACM Press, 1999.

[CT97] L. Cabbibo and R. Torlone, "Querying Multidimesional Databases", *6th International Workshop on Database Programming Languages (DBPL6)*, 1997.

[DJLS96] S. Dar, H.V.Jagadish, A. Levy, D. Srivastava. Answering queries with aggregation using views. In Proc. of *22nd International Conference on Very Large Data Bases (VLDB)*, Mumbai, India, 1996.

[Ende72] H.B. Enderton, A Mathematical Introduction to Logic. Academic Press, 1972.

[GBLP96] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tabs, and Sub-Totals. *Proceedings of the 12th International Conference on Data Engineering (ICDE)*, New Orleans, February 1996.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass, "Aggregate query processing in data warehouses", *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB),* Zurich, Switzerland, Morgan Kaufmann Publishers, August 1995.

[GJJ97] M. Gebhardt, M Jarke and S. Jacobs, "A toolkit for negotiation support on multi-dimensional data", *Proceedings of ACM SIGMOD International Conference on Management of Data*. Tucson, Arizona, 1997.

[GL97] M. Gyssens, L.V.S. Lakshmanan, "A Foundation for Multi-Dimensional Databases", *Proceedings of the 23rd International Conference on Very Large Databases (VLDB)*, Athens, August 1997.M. Gyssens, L.V.S. Lakshmanan. A Foundation for Multi-Dimensional Databases.

[GSW96] S. Guo, W. Sun, M. Weiss. Solving Satisfiability and Implication Problems in Database Systems. ACM Transactions on Database Systems, Vol. 21, No. 2, June 1996.

[Gupt97] H. Gupta: Selection of Views to Materialize in a Data Warehouse. In *Proceedings of the 6th International Conference on Database Theory (ICDT-97),* Delfi, Greece, 1997

[Lehn98] W. Lehner, "Modeling Large Scale OLAP Scenarios", *Proceedings of the 6th International Conference of Extending Database Technology (EDBT-98)*, 1998.

[LMSS95] A. Levy, A. Mendelzon, Y. Sagiv, D. Srivastava: Answering Queries Using Views. *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 1995.

[LS97] H. Lenz, A. Shoshani. Summarizability in OLAP and Statistical databases. In *Proceedings of 9th International Conference on Scientific and Statistical Database Management (SSDBM)*, 1997.

[LSTV99] S.Ligoudistianos, T.Sellis, D.Theodoratos, and Y.Vassiliou. Heuristic Algorithms for Designing the Data Warehouse with SPJ Views. *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery, (DaWaK)*, Lecture Notes in Computer Science, Vol. 1676, Springer, 1999.

[LW96] C. Li, X. Sean Wang, "A Data Model for Supporting On-Line Analytical Processing", *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 1996.

[Meta97] Metadata Coalition: Metadata Interchange Specification (MDIS v. 1.1). http://www.metadata.org/standards/toc.html 1997.

[MS98] Microsoft Corp. OLEDB for OLAP February 1998. Available at http://www.microsoft.com/data/oledb/olap/

[NSS98] W.Nutt, Y.Sagiv, S. Surin. Deciding Equivalences among Aggregate Queries. In Proc. *17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS),* Seattle, USA, 1998

[OLAP97] OLAP Council. The APB-1 Benchmark. 1997. Available at http://www.olapcouncil.org/research/bmarkly.htm

[ThLS99] D.Theodoratos, S.Ligoudistianos, and T.Sellis. Designing the Global DW with SPJ Queries. In Proc. *of the 11th Conference on Advanced Information Systems Engineering (CAiSE)*, pages 180--194, June 1999.

[TPC98] TPC: TPC Benchmark H and TPC Benchmark R. Transaction Processing Council. June 1999. Available at http://www.tpc.org/

[Ullm89] J. Ullman. "Principles of Database and Knowledge-Base Systems. Volume II: The New Technologies. Computer Science Press. 1989.

[Ullm97] J.D. Ullman, "Information integration using logical views", *Proceedings of the 6th International Conference on Database Theory (ICDT-97)*, Lecture Notes in Computer Science, pp. 19-40. Springer-Verlag, 1997.

[VaSe99] P. Vassiliadis, T. Sellis. "A Survey on Logical Models for OLAP Databases". *SIGMOD Record,* vol. 28, no. 4, December 1999.

[Vass98] P. Vassiliadis, "Modeling Multidimensional Databases, Cubes and Cube Operations", *Proceedings of 10th International Conference on Scientific and Statistical Database Management (SSDBM)*, Capri, Italy, July 1998.

[YaLa85] P. Larson, H. Z. Yang: Computing Queries from Derived Relations. In Proc. of VLDB 1985. *Proceedings of the 11th International Conference on Very Large Data Bases (VLDB),* Stockholm, Sweden, Morgan Kaufmann Publishers, August 1985.

Appendix: Proofs of all the Theorems

Proof of Theorem 2.1.

Soundness. If we consider a detailed data set DS^0 under the schema $S^0=[L_1^0,\dots,L_n^0,M_1^0,\dots,M_k^0]$, the valid cubes that can be defined over DS^0 must respect the following requirements:

C1. Their selection condition must be a detailed selection condition applicable to DS^0.

C2. All their levels must belong to the same dimensions with the respective levels of DS^0 (which also implies that they have the same number of dimensions).

C3. The number of their measures must be less or equal to the number of the measures of DS^0.

C4. All their measures must be produced from the application of `{sum,max,min,count}` to the respective measures of DS^0.

One can easily see that the results of all operations respect the constraints (C1) - (C4).

Completeness. Consider a detailed data set DS^0 under the schema $S^0=[L_1^0,\dots,L_n^0,M_1^0,\dots,M_k^0]$. Let also c be an arbitrary cube which can be defined over DS^0:

$$c=(DS^0,\varphi,[L_1,\dots,L_n,M_1,\dots,M_m],[agg_1(M_1^0),\dots,agg_m(M_m^0)]).$$

We can compute c as follows:

1. First, we construct the detailed cube. Although we use the aggregate function `sum`, one can use any other aggregation function at this stage:

$$c^0=(DS^0,true,[L_1^0,\dots,L_n^0,M_1^0,\dots,M_k^0],[sum(M_1^0),\dots,sum(M_k^0)])$$

2. Secondly, we split the non-relevant measures (which without loss of generality can be considered to be the last $k-m$ measures).

$$c^1=\pi_{M_{m+1}}(\pi_{M_{M_{m+2}}}(\dots(\pi_{M_k}(c^0))))=(DS^0,true,[L_1^0,\dots,L_n^0,M_1^0,\dots,M_m^0],[sum(M_1^0),\dots,sum(M_m^0)])$$

3. Then, we apply a selection operator to c^1 and we obtain a cube c^2:

$$c^2=\sigma_\varphi(c^1)=(DS^0,\varphi,[L_1^0,\dots,L_n^0,M_1^0,\dots,M_m^0],[sum(M_1^0),\dots,sum(M_m^0)])$$

4. Finally, we navigate to the levels of the schema of c.

$$nav(c^2,[L_1,\dots,L_n,M_1,\dots,M_m],[agg_1(M_1^0),\dots,agg_m(M_m^0)]]=$$

$$(DS^0,\varphi,[L_1,\dots,L_n,M_1,\dots,M_m],[agg_1(M_1^0),\dots,agg_m(M_m^0)])\equiv c. \blacksquare$$

Proof of Theorem 3.1.

Soundness. To show the axioms sound, we must show that they infer only true statements. In other words, we must show that they hold in each model. `A1-A8` are already proved to be sound in [Ullm89]. `A9-A10` are simple, well-known, properties of the monotone function f.

Completeness. We will extend the proof of [Ullm89]. Let $\mathbf{U}$ be a finite set of variables. Let $\mathbf{S}$ be a set of inequalities, involving a subset of $\mathbf{U}$, namely $\mathbf{V}$, and a finite set of variables $\mathbf{V'}$, such that all the variables participating in $\mathbf{V'}$, are of the form $\mathbf{V'}=\{f(X), X\in\mathbf{U}\}$. Let $\mathbf{U'}$ also be a finite set of "dummy" variables, having the same size with $\mathbf{U}$. We restrict $\mathbf{V}$ and $\mathbf{V'}$, so that we cannot derive from $\mathbf{S}$, using A1 through A15, inequalities of the form $X<X$, $X\leq Y \wedge Y\leq X$, $f(X)<f(X)$, $f(X)\theta Y$, $X\theta f(Y)$, $f(X)=f(Y)$ and $f(X)\leq f(Y)$ for every pair of distinct variables X and Y.

Suppose also that $\mathbf{S}^+$ is the set of inequalities that follow logically, using the axioms A1 to A15. We will refer to $\mathbf{S}^+$ as the *closure* of $\mathbf{S}$. If $\mathbf{S}^+$ is complete, every inequality $X\theta Y$ (respectively $f(X)\theta f(Y)$) not in $\mathbf{S}^+$, has some assignment of integers to the variables of $\mathbf{U}$ that makes every inequality in $\mathbf{S}^+$ true, but $X\theta Y$ (respectively $f(X)\theta f(Y)$) false.

We initialise $\mathbf{S}^+$ to be identical to $\mathbf{S}$. As a first step, we search the set $\mathbf{S}^+$ for pairs of inequalities of the form $f(X)\leq f(Y)$ and $f(Y)\leq f(X)$. We replace any occurrence of any of these inequalities in $\mathbf{S}^+$, with the single statement $f(X)=f(Y)$. After this step, one can be certain that if any inequality of the form $f(X)\leq f(Y)$ is found in $\mathbf{S}^+$, then no inequality of the form $f(Y)\leq f(X)$ is also found in $\mathbf{S}^+$.

Suppose also that $\mathbf{G}$ is a set, comprising of sets of variables. For each set of variables $\{f(X_1), f(X_2), \ldots, f(X_k)\}$ such that $f(X_1)=f(X_2)=\ldots=f(X_k)$, we introduce a new set in $\mathbf{G}$. Each set of this kind will be called *group* in the sequel. All the aforementioned transformations and set instantiations can be done in finite time. Using A13 and A15 we can easily prove that the sets of $\mathbf{G}$ are disjoint. Moreover, if $\mathbf{G}=\{g_1, g_2, \ldots\}$ and $A\in g_1, B\in g_2$, then:

- if $A<B$ then all members of g_1 are smaller than all member of g_2 (by A14 and A3);
- if $A\leq B$ then all members of g_1 are smaller or equal than all member of g_2 (by A15) and there does not exist $A'\in g_1, B'\in g_2$, such that $B'\leq A'$, because then the two sets would not be disjoint;
- if $A\neq B$, then all members of g_1 are different from all the members of g_2 (by A14).

In the sequel, we will often need to group and/or order the variables of $\mathbf{U}$. The result of this process will be referred to as $\mathbf{G}$.

In order to prove the completeness of $\mathbf{S}^+$, we must show that every inequality $X\theta Y$ (respectively $f(X)\theta f(Y)$) not in $\mathbf{S}^+$, has some assignment of integers to the variables of $\mathbf{U}$ that makes every inequality in $\mathbf{S}^+$ true, but $X\theta Y$ (respectively $f(X)\theta f(Y)$) false. Depending on the type of the inequality we consider the following cases:

Case 1: $X\leq Y$. We need to construct an assignment that satisfies $\mathbf{S}^+$ but makes $Y<X$ true. Let $\mathbf{A}$ be a set of variables A such that $X\leq A\in\mathbf{S}^+$ and $\mathbf{B}$ a set of variables B such that $B\leq Y\in\mathbf{S}^+$. Let also $\mathbf{C}=\mathbf{U}-\mathbf{A}-\mathbf{B}$. We will refer to $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ as *blocks*. In general, if $A\in\mathbf{A}$, $B\in\mathbf{B}$, and $C\in\mathbf{C}$, we can be sure that neither $C\leq B$ nor $A\leq C$ holds. Also, the sets $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are disjoint. ($\mathbf{C}$ is disjoint from the other two sets by definition. $\mathbf{A}$ and $\mathbf{B}$ are also disjoint or else $\exists K\in\mathbf{A}, \mathbf{B}$, s.t. $X\leq K, K\leq Y\in\mathbf{S}^+$, which implies $X\leq Y\in\mathbf{S}^+$, contrary to our initial assumption). Note also that all variables of the form $f(X)$ are found in $\mathbf{C}$, since we do not allow inequalities of the form $f(X)\theta Y$.

Now, we topologically sort **A**, **B** and **C** with respect to the order $\leq$. Remember that after the initial transformation of $\mathbf{S}^+$ we cannot derive both $A_1 \leq A_2$ and $A_2 \leq A_1$ for two distinct variables. For **A** and **B**, which do not involve variables of the form f(X), the sorting is straightforward. For **C** we use the following trick: all the variables f(U) and f(V) such that f(U)=f(V) (i.e. belonging in the same group in **G**), are replaced from a single dummy variable, just for the sake of the topological ordering). After this transformation the topological ordering is feasible, since we cannot derive both $f(U) \leq f(V)$ and $f(V) \leq f(U)$ for two distinct variables. After the topological ordering is performed we replace the dummy variable with the respective variables, placed in consecutive positions. Next, we order all the variables in **U**, as follows: First we list all the elements of **B** in topological order, then all the elements of **C** in order and finally all the elements of **A** in order. After that, we assign distinct integers 1, 2, ... to the variables in this order, except for variables belonging to the same group, which are assigned the same integer. We call the integer assigned to a variable U, through this assignment, a(U).

So forth, we have managed that X is given a larger value than Y, so $a(X) \leq a(Y)$ does not hold. Now, we must show that all the inequalities in $\mathbf{S}^+$ hold in this assignment. We will use the variable names U, V for the members of **V** and f(U), f(V) for the members of **V'**. Depending on the form of inequalities we consider the following cases:

a. $U \neq V \in \mathbf{S}^+$. Since no variables of this form are assigned to the same integer, all $\neq$ inequalities hold.

b. $U \leq V \in \mathbf{S}^+$. We have the following possible combinations for U and V:

b_1. U, V are the same variable. Then, $a(U) \leq a(V)$ holds.

b_2. U, V are in the same block. Then, $a(U) \leq a(V)$ holds, since each block is topologically ordered.

b_3. $U \in \mathbf{B}$, $V \in \mathbf{C}$ (or **A**). Then, $a(U) \leq a(V)$ holds, since in the final order, everything in **B** precedes everything else in the other blocks.

b_4. $U \in \mathbf{C}$, $V \in \mathbf{A}$. Then, $a(U) \leq a(V)$ holds for the same reason.

b_5. $U \in \mathbf{A}$ then V must also be in **A** (or else A7 is violated) so b_1 holds. Moreover, if $U \in \mathbf{C}$ then V must be in **A** so b_4 holds.

All the above are summarised in the following table.

U↓ V→	B	C	A
B	b_2	b_3	b_3
C	b_5	b_2	b_4
A	b_5	b_5	b_2

c. $U < V \in \mathbf{S}^+$. Obviously, U, V are not the same variable. The argument given for $U \leq V$ can also be given in this case (b_2-b_5). Thus $U < V$ is satisfied by the proposed assignment.

d. $f(U) \neq f(V) \in \mathbf{S}^+$. All these variables are in **C**. Due to the nature of the assignment and the fact that only equal variables are assigned the same integer, the inequality holds.

e. $f(U) \leq f(V) \in \mathbf{S}^+$. The assignment respects the order, due to its nature. Remember that due to the "cleaning" transformations, inequalities of the form $f(V) \leq f(U)$ and $f(V) = f(U)$ do not exist in $\mathbf{S}^+$at the same time.

f. $f(U) < f(V) \in \mathbf{S}^+$. Obviously, $f(U) \neq f(V)$ (i.e., they are not the same variable). The argument for case **e** can be reused.

g. $f(U) = f(V) \in \mathbf{S}^+$. Due to the nature of the assignment, the variables which are equal to each other are assigned the same integer.

Since all the possible combinations of expressions that can be found in $\mathbf{S}^+$ respect the assignment and $a(Y) < a(X)$ then we conclude that the assignment satisfies $\mathbf{S}^+$ but not $X \leq Y$.

Case 2: Suppose that $X \neq Y$ is not in $\mathbf{S}^+$. We must find an assignment such that $a(X) \neq a(Y)$ is false and all the other assignments in $\mathbf{S}^+$ are true. [Ullm89] observes that there cannot be two distinct variables Z, W such that $X \leq Z$ and $Z \leq Y$ and $Y \leq W$ and $W \leq X$. Consequently, there is an order between X, Y and without loss of generality we can assume that $X \leq Y$. We define the following blocks: Let $\mathbf{Z}$ be a set containing all variables Z such that $X \leq Z \leq Y \in \mathbf{S}^+$, $\mathbf{D}$ be the set containing X, Y and all the variables from set $\mathbf{Z}$, $\mathbf{A}$ be the set of variables A such that $D \leq A \in \mathbf{S}^+$, for any $D \in \mathbf{D}$, but A is not itself in $\mathbf{D}$, $\mathbf{B}$ is the set of variables B such that $B \leq D$ is in $\mathbf{S}^+$, for any $D \in \mathbf{D}$, but B is not itself in $\mathbf{D}$ and $\mathbf{C}$ be the set of variables that remain. Sets $\mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{A}$ are disjoint. ($\mathbf{C}$ is disjoint from the rest of the sets by definition. $\mathbf{A}$ and $\mathbf{B}$ are disjoint from $\mathbf{D}$, by definition too. $\mathbf{A}$ and $\mathbf{B}$ are disjoint, or else $\exists K \in \mathbf{A}, \mathbf{B}$ and $D_1, D_2 \in \mathbf{D}$ such that, $D_1 \leq K, K \leq D_2 \in \mathbf{S}^+$, which implies $X \leq D_1 \leq K \leq D_2 \leq Y \in \mathbf{S}^+$, meaning that $K \in \mathbf{Z}$ and consequently $K \in \mathbf{D}$, contrary to our initial assumption). If $A \in \mathbf{A}$, $B \in \mathbf{B}$ and $C \in \mathbf{C}$ then it is possible that $B \leq C \leq A$ and $B \leq D \leq A$. Moreover, the following do not hold: $A \leq C, C \leq B, A \leq D, D \leq B$. For example, $A \leq C$ cannot hold, because if it did, then $\exists D \in \mathbf{D}$ such that, $D \leq A \leq C \in \mathbf{S}^+$, implying that $C \in \mathbf{A}$, which cannot hold. The rest of the cases can be proved similarly. Again all the variables of the form $f(U)$ are found in $\mathbf{C}$.

We topologically order all these sets of variables with respect to the order $\leq$, using the methodology of **Case 1**. We combine the orders in the sequence $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, $\mathbf{A}$. We assign distinct integers to all variables, except (a) for the variables of $\mathbf{D}$ and (b) for the variables belonging to the same set in $\mathbf{G}$. In both these cases, each set of variables is given the same integer. Surely, $X \neq Y$ is not satisfied by this assignment, since $a(X) = a(Y)$. We will prove that all the inequalities in $\mathbf{S}^+$ are satisfied. We consider again the following seven cases for inequalities in $\mathbf{S}^+$:

a. $U \neq V \in \mathbf{S}^+$. Since no variables of this form are assigned to the same integer, except for variables belonging to $\mathbf{D}$, all these inequalities hold. For the case where both U, V belong to $\mathbf{D}$, we can prove that it is not possible that $U \neq V \in \mathbf{S}^+$. Suppose that $\exists Z, W \in \mathbf{D}$, such that $X \leq Z \leq Y, X \leq W \leq Y, Z \neq W \in \mathbf{S}^+$. Then $X \neq Y$ is implied (by A8), which contradicts our hypothesis. Consequently, there are no $\neq$ inequalities between members of $\mathbf{D}$.

b. $U \le V \in \mathbf{S}^+$. We have the following possible combinations for U and V:

b_1. U, V are the same variable. Then, $a(U) \le a(V)$ holds.

b_2. U, V are in the same block. Then, $a(U) \le a(V)$ holds, since each block is topologically ordered and the topological order respects the inequality for this kind of variables.

b_3. $U \in \mathbf{B}$ and $V \in \mathbf{C}, \mathbf{D}$ or $\mathbf{A}$. Then, $a(U) \le a(V)$ holds, since in the final order, everything in $\mathbf{B}$ precedes everything else in the other blocks.

b_4. $U \in \mathbf{C}$ and $V \in \mathbf{D}$ or $\mathbf{A}$. For the same reason, $a(U) \le a(V)$ holds.

b_5. $U \in \mathbf{D}$ and $V \in \mathbf{A}$. For the same reason, $a(U) \le a(V)$ holds.

b_6. $U \in \mathbf{A}$ then V must be in $\mathbf{A}$ (or else A7 is violated). Moreover, if $U \in \mathbf{C}$ then V can not be in $\mathbf{B}$.

b_7. $U \in \mathbf{D}$ then $V \in \mathbf{D}, \mathbf{A}$ and not in $\mathbf{B}$ or $\mathbf{C}$. This holds because if $V \in \mathbf{B}$ then $D \le B$ which does not hold. If, on the other hand, $V \in \mathbf{C}$ then $X \le U \le Y$ and $U \le V$, implying that $\exists D \in \mathbf{D}$, such that $D \le V$, i.e., $V \in \mathbf{A}$, which does not hold either.

c. $U < V \in \mathbf{S}^+$. Obviously, U, V are not the same variable. The argument given for $U \le V$ can also be given in this case too (b_2-b_7). Thus $U < V$ is satisfied by the proposed assignment.

d. $f(U) \ne f(V) \in \mathbf{S}^+$. All these variables are in $\mathbf{C}$. Due to the nature of the assignment and the fact that only equal variables are assigned the same integer, the inequality holds.

e. $f(U) \le f(V) \in \mathbf{S}^+$. The assignment respects the order, due to its nature. Remember that due to the "cleaning" transformations, inequalities of the form $f(V) \le f(U)$ or $f(V) = f(U)$ do not exist in $\mathbf{S}^+$.

f. $f(U) < f(V) \in \mathbf{S}^+$. Obviously, $f(U) \ne f(V)$ (i.e., they are not the same variable). The argument for case **e** can be reused.

g. $f(U) = f(V) \in \mathbf{S}^+$. Due to the nature of the assignment, the variables which are equal to each other are assigned the same integer.

Since all the possible combinations of expressions that can be found in $\mathbf{S}^+$, respect the assignment and $a(Y) = a(X)$ then we conclude that the assignment satisfies $\mathbf{S}^+$ but not $X \ne Y$.

Case 3: Suppose that $X < Y$ is not in $\mathbf{S}^+$. We must find an assignment such that $X < Y$ is false and all the other assignments in $\mathbf{S}^+$ are true. If both $X \le Y$ and $X \ne Y$ are in $\mathbf{S}^+$, then, by A4, $X < Y$ is also in $\mathbf{S}^+$, contrary to the initial assumption for **Case 3**. Consequently, for $X < Y$ not to be in $\mathbf{S}^+$, at least one of the following two cases must hold:

a. $X \le Y \notin \mathbf{S}^+$. Then we can use the assignment of **Case 1**; it satisfies $\mathbf{S}^+$ and proves that $X < Y$ is false.

b. $X \ne Y \notin \mathbf{S}^+$. Then we can use the assignment of **Case 2**; it also satisfies $\mathbf{S}^+$ and proves that $X < Y$ is false.

In both cases, $a(X) < a(Y)$ is false and all the other assignments in $\mathbf{S}^+$ are true.

Case 4: Suppose that $f(X) \le f(Y)$ is not in $\mathbf{S}^+$. We need to construct an assignment that satisfies $\mathbf{S}^+$ but makes $a(f(Y)) < a(f(X))$ true. Let $\mathbf{A}$ be a set of variables A such that $f(X) \le f(A)$ is in $\mathbf{S}^+$ and $\mathbf{B}$ a set of variables $\mathtt{B}$ such that $\mathtt{f(B)}\leq\mathtt{f(Y)}$ is also in $\mathbf{S}^+$. Let also $\mathbf{C}=\mathbf{U}-\mathbf{A}-\mathbf{B}$. In general, if $\mathtt{f(A)}\in\mathbf{A}$, $\mathtt{f(B)}\in\mathbf{B}$, and $\mathtt{f(C)}\in\mathbf{C}$ we can be sure that neither $\mathtt{f(C)}\leq\mathtt{f(B)}$, nor $\mathtt{f(A)}\leq\mathtt{f(C)}$ hold. Also, the sets $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are disjoint (similarly with **Case 1**). Note that all variables belonging to $\mathbf{V}$ (i.e., of the form $\mathtt{U}$) are found in $\mathbf{C}$, since we do not allow inequalities of the form $\mathtt{f(X)}\theta\mathtt{Y}$. Moreover, one can show that if a variable $\mathtt{f(U)}$ belongs to a block and there exists a variable $\mathtt{f(V)}$, such that $\mathtt{f(U)=f(V)}$, then both variables are found in the same block. This holds for all the three blocks $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. Moreover, $\mathtt{f(X)}$, $\mathtt{f(Y)}$ are allowed to be found in different blocks, since if $\mathtt{f(U)=f(V)}$ is found in $\mathbf{S}^+$, this would imply that $\mathtt{f(X)}\leq\mathtt{f(Y)}$ (A15) is also found in $\mathbf{S}^+$, which we have already assumed that is not true.

Now, we topologically sort $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ with respect to the order $\leq$. We use the following trick: all the variables $\mathtt{f(U)}$ and $\mathtt{f(V)}$ such that $\mathtt{f(U)=f(V)}$, i.e., belonging in the same group in $\mathbf{G}$ are replaced from a single dummy variable, just for the sake of the topological ordering. Now, the topological ordering is feasible, since we cannot derive both $\mathtt{f(U)}\leq\mathtt{f(V)}$ and $\mathtt{f(V)}\leq\mathtt{f(U)}$ for two distinct variables. After the topological ordering is performed we replace the dummy variable with the respective variables, placed in consecutive positions. Next, we order all the variables in $\mathbf{U}$, as follows: First we list all the elements of $\mathbf{B}$ in order, then all the elements of $\mathbf{C}$ in order and finally all the elements of $\mathbf{A}$ in order. After that, we assign distinct integers 1, 2, ... to the variables in this order, except for variables belonging to the same group, which are assigned the same integer. We call the integer assigned to a variable $\mathtt{U}$, through this assignment, $\mathtt{a(U)}$.

So forth, we have managed that $\mathtt{f(X)}$ is given a larger value than $\mathtt{f(Y)}$, so $\mathtt{f(X)}\leq\mathtt{f(Y)}$ does not hold (remember that $\mathtt{f(X)}$, $\mathtt{f(Y)}$ cannot be related with equality, so they are assigned different numbers). Now, we must show that all the other inequalities in $\mathbf{S}^+$ hold in this assignment. We consider the following cases.

a. $\mathtt{U}\neq\mathtt{V}\in\mathbf{S}^+$. Since no variables of this form are assigned to the same integer, all $\neq$ inequalities hold.

b. $\mathtt{U}\leq\mathtt{V}\in\mathbf{S}^+$. All these variable belong to $\mathbf{C}$, there are no equalities between them and the topological ordering assigns different integers to different variables: consequently, the assignment holds.

c. $\mathtt{U}<\mathtt{V}\in\mathbf{S}^+$. Obviously, $\mathtt{U}$ and $\mathtt{V}$ are not the same variable. The argument given for $\mathtt{U}\leq\mathtt{V}$ can also be given in this case. Thus $\mathtt{U}<\mathtt{V}$ is satisfied by the proposed assignment.

d. $\mathtt{f(U)}\leq\mathtt{f(V)}\in\mathbf{S}^+$. All variables of the same group in $\mathbf{G}$, belong to the same block. We consider the following cases:

- b_1. $\mathtt{f(U)}$ and $\mathtt{f(V)}$ are in the same block. Remember that $\mathtt{f(X)=f(Y)}$ is not in $\mathbf{S}^+$ and that only equal variables are assigned the same integer. Then, $\mathtt{a(f(U))}\leq\mathtt{a(f(V))}$ holds, since each block is topologically ordered and the topological order respects the inequality for this kind of variables.
- b_2. $\mathtt{f(U)}\in\mathbf{B}$ and $\mathtt{f(V)}\in\mathbf{C}$ or $\mathbf{A}$. Then, $\mathtt{a(f(U))}\leq\mathtt{a(f(V))}$ holds, since in the final order, everything in $\mathbf{B}$ precedes everything else in the other blocks.
- b_3. $\mathtt{f(U)}\in\mathbf{C}$ and $\mathtt{f(V)}\in\mathbf{A}$. Then, for the same reason, $\mathtt{a(f(U))}\leq\mathtt{a(f(V))}$ holds.

b_4. $f(U) \in \mathbf{A}$. Then, $f(V) \in \mathbf{A}$ (or else A7 is violated), so $f(V)$ cannot be in $\mathbf{B}$, $\mathbf{C}$. The same happens if $f(U) \in \mathbf{C}$ and $f(V) \in \mathbf{B}$.

e. $f(U) \neq f(V) \in \mathbf{S}^+$. These two variables do not belong to the same group in $\mathbf{G}$, thus they are assigned different integers.

f. $f(U) < f(V) \in \mathbf{S}^+$. Obviously, $f(U) \neq f(V)$ (i.e., they are not the same variable). The argument for **d** can be reused.

g. $f(U) = f(V) \in \mathbf{S}^+$. Due to the nature of the assignment, the variables which are equal to each other are assigned the same integer.

Since all the possible combinations of expressions that can be found in $\mathbf{S}^+$, respect the assignment and $a(f(Y)) < a(f(X))$ then we conclude that the assignment satisfies $\mathbf{S}^+$ but not $f(X) \leq f(Y)$.

Case 5: Suppose that $f(X) \neq f(Y) \notin \mathbf{S}^+$. We need to construct an assignment that satisfies $\mathbf{S}^+$ but makes $a(f(Y)) \neq a(f(X))$ false. First, note that since $f(X) \neq f(Y)$ is not in $\mathbf{S}^+$, then (by A5 and A3) also $f(X) < f(Y)$ and $f(Y) < f(X)$ are not in $\mathbf{S}^+$. Since $f(Y) < f(X)$ is not in $\mathbf{S}^+$, we can construct the following set $\mathbf{D}$:

$$\mathbf{D} = \{f(D) : \{f(X) \leq f(D) \leq f(Y)\} \cup \{f(D) = f(X)\} \cup \{f(D) = f(Y)\}\}$$

Note that the definition of $\mathbf{D}$ is legitimate. There are three cases: (a) $f(X) \leq f(Y) \in \mathbf{S}^+$, (b) $f(Y) \leq f(X) \in \mathbf{S}^+$, (c) none of the previous two. In the first case the definition of $\mathbf{D}$ is as previously mentioned. In the second case, obviously $f(X), f(Y)$ are used in reverse order. In the third case, the set $\{f(X) \leq f(D) \leq f(Y)\}$ is empty. Without loss of generality, we choose that the first case holds.

We construct the following sets: Let $\mathbf{A}$ be the set of variables $f(A)$ such that $f(D) \leq f(A) \in \mathbf{S}^+$, for some $f(D) \in \mathbf{D}$, but $f(A)$ is not itself in $\mathbf{D}$, let $\mathbf{B}$ is the set of variables $f(B)$ such that $f(B) \leq f(D) \in \mathbf{S}^+$, for some $f(D) \in \mathbf{D}$, but $f(B)$ is not itself in $\mathbf{D}$ and let $\mathbf{C}$ be the set of remaining variables. The sets $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{A}$ are disjoint. $\mathbf{C}$ is disjoint from the rest of the sets by definition. $\mathbf{A}$ and $\mathbf{B}$ are disjoint from $\mathbf{D}$, by definition too. $\mathbf{A}$ and $\mathbf{B}$ are disjoint, or else $\exists f(K) \in \mathbf{A},\mathbf{B}$, $f(D) \in \mathbf{D}$ such that, $f(D) \leq f(K), f(K) \leq f(D) \in \mathbf{S}^+$, which implies $f(K) = f(D)$, meaning that $f(K) \in \mathbf{D}$, which cannot hold. Moreover if $f(B) \in \mathbf{A}$, $f(D) \in \mathbf{D}$, $f(B) \in \mathbf{B}$ and $f(C) \in \mathbf{C}$ then $f(B) \leq f(C) \leq f(A)$ and $f(B) \leq f(D) \leq f(A)$ can possibly hold and $f(A) \leq f(C)$, $f(C) \leq f(B)$, $f(A) \leq f(D)$ and $f(D) \leq f(B)$ do not hold. Again all the variables of the form U are found in $\mathbf{C}$.

We topologically order these sets of variables with respect to the order $\leq$, using the methodology of **Case 1**. We combine the orders in the sequence $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{A}$. We assign distinct integers to all variables, except (a) for the variables of $\mathbf{D}$ and (b) for the variables belonging to the same set in $\mathbf{G}$. In both these cases, each set of variables is given the same integer.

Surely, $a(X) \neq a(Y)$ is not satisfied by this assignment. We will prove that all the inequalities in $\mathbf{S^+}$ are satisfied. We consider again the following cases:

a. $U \neq V \in \mathbf{S^+}$. Since no variables of this form are assigned to the same integer, all such $\neq$ inequalities hold.

b. $U \leq V \in \mathbf{S^+}$. All these variable belong to $\mathbf{C}$, there are no equalities between them and the topological ordering assigns different integers to different variables: consequently, the assignment holds.

c. $U < V \in \mathbf{S^+}$. Obviously, U, V are not the same variable. The argument given for $U \leq V$ can also be given in this case. Thus $U < V$ is satisfied by the proposed assignment.

d. $f(U) \neq f(V) \in \mathbf{S^+}$. By default any variables that do not belong to the same group in $\mathbf{G}$, are assigned different integers. The only problem could happen with variables belonging to $\mathbf{D}$. Similarly with **Case 2**, we can prove that there are no variables $f(U) \neq f(V)$ such that $f(X) \leq f(U) \leq f(Y)$, $f(X) \leq f(V) \leq f(Y) \in \mathbf{S^+}$. Moreover, if there is a variable belonging to the same group with either $f(X)$ or $f(Y)$, then there is no possibility for a $\neq$ inequality to hold for it, since then $f(X) \neq f(Y)$ would belong to $\mathbf{S^+}$.

e. $f(U) \leq f(V) \in \mathbf{S^+}$. For any pair of variables the inequality for the assignment holds (as shown in previous cases). For the case of two variables belonging to $\mathbf{D}$, then $a(f(U)) \leq a(f(V))$ holds, since $a(f(U)) = a(f(V))$ holds.

f. $f(U) < f(V) \in \mathbf{S^+}$. As with the case of **e**, for any pair of variables the inequality for the assignment holds (as shown in previous cases). For the case of two variables belonging to $\mathbf{D}$, it is not possible that such an inequality holds (or else $f(U) < f(V)$ would imply $f(U) \neq f(V) \in \mathbf{S^+}$).

g. $f(U) = f(V) \in \mathbf{S^+}$. Due to the nature of the assignment, the variables which are equal to each other are assigned the same integer.

Case 6: Suppose that $f(X) < f(Y)$ is not in $\mathbf{S^+}$. Similarly with **Case 3**, we must find an assignment such that $a(f(X)) < a(f(Y))$ is false and all the other assignments in $\mathbf{S^+}$ are true. If both $f(X) \leq f(Y)$ and $f(X) \neq f(Y)$ are in $\mathbf{S^+}$, then, by A4, $f(X) < f(Y)$ is also in $\mathbf{S^+}$, contrary to the initial assumption for **Case 6**. Consequently, for $f(X) < f(Y)$ not to be in $\mathbf{S^+}$, at least one of the following two cases must hold:

a. $f(X) \leq f(Y) \notin \mathbf{S^+}$. Then we can use the assignment of **Case 4**; it satisfies $\mathbf{S^+}$ and proves that $f(X) < f(Y)$ is false.

b. $f(X) \neq f(Y) \notin \mathbf{S^+}$. Then we can use the assignment of **Case 5**; it also satisfies $\mathbf{S^+}$ and proves that $f(X) < f(Y)$ is false.

In both cases, $a(f(X)) < a(f(Y))$ is false and all the other assignments in $\mathbf{S^+}$ are true.

Case 7: Suppose that $f(X) = f(Y) \notin \mathbf{S^+}$. We need to construct an assignment that satisfies $\mathbf{S^+}$ but makes $a(f(Y)) = a(f(X))$ true. Since $f(X) = f(Y) \notin \mathbf{S^+}$, we can deduce that only one of the following three cases can hold: (a) $f(X) \leq f(Y) \in \mathbf{S^+}$, (b) $f(Y) \leq f(X) \in \mathbf{S^+}$, (c) none of the previous two. For the second and the third case we can use directly the assignment of **Case 4**, which produces $a(f(Y)) < a(f(X))$ (implying that

$(f(Y)) \neq a(f(X)))$ and respects all the inequalities found in $\mathbf{S}^+$, at the same time. For the first case, we can use the assignment of **Case 4** with reverse roles for $f(Y)$ and $f(X)$. ■

In order to check whether one set of inequalities T follows from an other S we compute the closure $\mathbf{S}^+$ by applying the axioms A1-A15 until they no longer generate any new inequalities. Then, we check whether T is a subset of $\mathbf{S}^+$.

Proof of Theorem 3.2.

We distinguish two cases for the theorem: (a) selection conditions involving only levels and (b) selection conditions involving atoms of the form $L_i\theta l$, $anc_{L^0}^{L}(L^0)\theta l$ or their equivalent ones..

Case 1: atoms involving only levels, i.e., of the form $L_i\theta L_j$, $L_i^{old}\prec L_i$, $L_j^{old}\prec L_j$ or their equivalent ones.

In **Lines 11-17** of Algorithm Cube_Usability, we transform the expression φ_{new} at the levels of the schema of c^{old}, and obtain the expression $\varphi_{n@o}$. To do so, we modify all the atoms of φ_{new} which are of the form $L_i\theta L_j$, $L_i^{old}\prec L_i$, $L_j^{old}\prec L_j$, to the equivalent form $anc_{L_i^0}^{L_i^{old}}(L_i^{old})\theta anc_{L_j^0}^{L_j^{old}}(L_j^{old})$. The equivalence holds because all the levels occurring in the expression are higher than the respective levels in the schema of c^{old}.

Line 18. The new selection condition $\varphi_{n@o}$ is equivalent to φ_{new}. Moreover since $\varphi_{new}\subseteq_{\mathbf{S}}\varphi_{old}$, we can deduce (a) $\varphi_{n@o}\subseteq_{\mathbf{S}}\varphi_{old}$, (b) $\varphi_{n@o}\wedge\varphi_{old}\equiv\varphi_{n@o}\equiv\varphi_{new}$. Thus, if we apply $\varphi_{n@o}$ to c^{old}, treating c^{old} as a data set, we have

$\sigma_{\varphi_{n@o}}(c^{old})=DS^1=(DS^0,\varphi_{n@o}\wedge\varphi_{old},[L_1^{old},\dots,L_n^{old},M_{old}],agg_{old}(M^0))\equiv(DS^0,\varphi_{new},[L_1^{old},\dots,L_n^{old},M_{old}],$ $agg_{old}(M^0))=c^1$

Line 19-20. Next, we aggregate on the tuples of c^1, so that we produce c^{new}. Note that we use c^1 and c^{old} as data sets and not as expressions over the detailed data set. The transformation is done in the following manner:

$$DS^2=\{x'\in Tup(L_1^{new},\dots,L_n^{new},M_{new})\mid \exists x\in c^1:\ x'[L_i^{new}]=anc_{L_i^{old}}^{L_i^{new}}(x[L_i^{old}]),1\le i\le n,$$

$$x'[M_{new}]=agg(\{f\mid\exists g\in c^1,x'[L_i^{new}]=\ anc_{L_i^{old}}^{L_i^{new}}(g[L_i^{old}]),1\le i\le n,f=g[M_{old}]\})\Leftrightarrow(1)$$

$$DS^2=\{x'\in Tup(L_1^{new},\dots,L_n^{new},M_{new})\mid \exists y\in\varphi_{new}(DS^0),x\in c^1,\ \ (Note:\varphi_{n@o}\wedge\varphi_{old}\equiv\varphi_{new})$$

$$x[L_i^{old}]=anc_{L_i^0}^{L_i^{old}}\ (y[L_i^0]),1\le i\le n,\ \ x'[L_i^{new}]=anc_{L_i^{old}}^{L_i^{new}}(x[L_i^{old}]),1\le i\le n,$$

$$x'[M_{new}]=\ agg(\{f\mid\exists g\in c^1,x'[L_i^{new}]=\ anc_{L_i^{old}}^{L_i^{new}}(g[L_i^{old}]),1\le i\le n,$$

$$f=agg_{old}(\{q\mid\exists z\in\varphi_{new}(DS^0),x[L_i^{old}]=anc_{L_i^0}^{L_i^{old}}\ (z[L_i^0]),1\le i\le n,q=z[M^0]\})\})\}\Leftrightarrow(2)$$

$$DS^2=\{x'\in Tup(L_1^{new},\dots,L_n^{new},M_{new})\mid \exists y\in\varphi_{new}(DS^0),x\in c^1,$$

$$x'[L_i^{new}]=anc_{L_i^{old}}^{L_i^{new}}(anc_{L_i^0}^{L_i^{old}}\ (y[L_i^0])),1\le i\le n,$$

$$x'[M_{new}]=agg(\{f\mid\exists g\in c^1,z\in\varphi_{new}(DS^0),$$

$$x'[L_i^{new}]=anc_{L_i^{old}}^{L_i^{new}}(anc_{L_i^0}^{L_i^{old}}(z[L_i^0])),1\le i\le n,f=agg_{old}(z[M^0])\})\} \Leftrightarrow (3)$$

$$DS^2=\{x'\in Tup(L_1^{new},\dots,L_n^{new},M_{new}) \mid \exists y\in\varphi_{new}(DS^0),$$

$$x'[L_i^{new}]=anc_{L_i^0}^{L_i^{new}}(y[L_i^0])),1\le i\le n,x'[M_{new}]=agg(\{agg_{old}(z[M^0]) \mid \exists z\in\varphi_{new}(DS^0),$$

$$x'[L_i^{new}]=anc_{L_i^0}^{L_i^{new}}(z[L_i^0])),1\le i\le n\})\} \Leftrightarrow (4)$$

$DS^2 = c^{new}$.

Discussion. The first equivalence (1) holds, since we have simply expanded the semantics for c^2, by incorporating the definition of c^1 into the formula. The second equivalence (2) obviously holds, since we just combine the atoms in the formula. The third equivalence (3) holds due to the transitivity of the ancestor function. The fourth equivalence (4) holds since we can always find a function agg, which applied over the function agg_{old} will produce the function agg_{new}. More specifically,

$$sum^{\circ}sum=sum, min^{\circ}min=min, max^{\circ}max=max, sum^{\circ}count=count.$$

Consequently, the rewriting operations do indeed produce the cube c^{new}.

Case 2: atoms involving levels and values, i.e., of the form $L_i\theta l$, $anc_{L^0}^{L}(L^0)\theta l$ or their equivalent ones. Let $\mathbf{L}$ be the schema of c^{new}.

Since $\varphi_{new}\subseteq_{\mathbf{L}}\varphi_{old}$ then there exist $\varphi'_{new},\varphi'_{old}$ produced from Algorithm Check_Atoms_Usability as conjunctions of dimension intervals, such that:

- $\varphi'_{new},\varphi'_{old}$ are conjunctions of dimension intervals. Let a_i^{new} be a DI covering dimension D_i and belonging to φ'_{new}. The respective notation is also used for the atoms of φ'_{old}.
- $\varphi'_{new}\subseteq_{\mathbf{L}}\varphi'_{old}$, i.e., for each atom a_i^{new} there exists an atom a_i^{old} such that $a_i^{new}\subseteq_{\mathbf{L}}a_i^{old}$ and of course $a_i^{new}\subseteq a_i^{old}$.
- $\varphi'_{new}\equiv\varphi_{new}$ and $\varphi'_{old}\equiv\varphi_{old}$.

The fact that $\varphi'_{new}\subseteq_{\mathbf{L}}\varphi'_{old}$ means that for each dimension D_i the domain of L_i^0 is partitioned with respect to the level L_i^{new}. If we call p_{ij}^{new} each partition created by φ'_{new}, with respect to level L_i^{new}, we have:

- $\bigcup_j p_{ij}^{new}\subseteq dom(L_i^0)$, $\bigcup_k p_{ik}^{old}\subseteq dom(L_i^0)$,
- for each p_{ij}^{new} there exists a p_{ik}^{old} such that $p_{ij}^{new}=p_{ik}^{old}$ and
- finally, if x_1,x_2 belong to the same p_{ij} then $anc_{L_i^0}^{L_i^{new}}(x_1)=anc_{L_i^0}^{L_i^{new}}(x_2)$. In other words, each partition is defined by the fact that its elements have a common ancestor at the level of L_i^{new}.

Due to the properties of the family of anc functions (i.e., disjointness of descendants and monotonicity) and the fact that $L_{old}\prec L_{new}$ we can claim that each partition defined with respect to the levels of c^{new} is the union of all the partitions that can be defined for the levels of c^{old}.

Notation: We will use the letter $\mathtt{p}$ to denote partitions defined from levels of $\mathtt{c}^{\mathtt{new}}$ and $\mathtt{q}$ for partitions defined from levels of $\mathtt{c}^{\mathtt{old}}$. The superscript in these symbols denotes the selection condition from which the partitions are derived.

For each $\mathtt{p}_{\mathtt{ij}}^{\mathtt{new}}$, there exist a set of $\mathtt{q}_{\mathtt{il}}^{\mathtt{new}}$such that

- $\mathtt{p}_{\mathtt{ij}}^{\mathtt{new}}=\bigcup_{\mathtt{l}}\mathtt{q}_{\mathtt{il}}^{\mathtt{new}}$ and
- for each $\mathtt{x}_1,\mathtt{x}_2$ belonging to the same $\mathtt{q}_{\mathtt{ij}}$ then $\mathtt{anc}_{\mathtt{L}_\mathtt{i}^0}^{\mathtt{L}_\mathtt{i}^{\mathtt{old}}}(\mathtt{x}_1)=\mathtt{anc}_{\mathtt{L}_\mathtt{i}^0}^{\mathtt{L}_\mathtt{i}^{\mathtt{old}}}(\mathtt{x}_2)$.

Exactly the same relationships hold for all the $\mathtt{p}_{\mathtt{ik}}^{\mathtt{old}}$.

For example, the partition of a particular year is the union of all the partitions for its months, if we consider $\mathtt{c}^{\mathtt{old}}$ to be at the month level).

The question arises, then, about the relationship between partitions and dimension intervals. We introduce the term *covers* between a DI $\mathtt{a}$ and a partition $\mathtt{p}$ if `Check_DI_Usability(p,a,L`$_{\mathtt{new}}$`)=true`, i.e., $\mathtt{p}$ is a subset of $\mathtt{a}$ and they have the proper marginal conditions. Consequently, for each $\mathtt{a}_\mathtt{i}^{\mathtt{new}}$ the following hold:

- there exists a set of partitions $\mathtt{p}_{\mathtt{ij}}^{\mathtt{new}}$, such that $\mathtt{a}_\mathtt{i}^{\mathtt{new}}=\bigcup_{\mathtt{j}}\mathtt{p}_{\mathtt{ij}}^{\mathtt{new}}$,
- there exists an atom $\mathtt{a}_\mathtt{i}^{\mathtt{old}}$ such that $\mathtt{a}_\mathtt{i}^{\mathtt{new}}\subseteq_{\mathbf{L}}\mathtt{a}_\mathtt{i}^{\mathtt{old}}$,
- for each $\mathtt{p}_{\mathtt{ij}}^{\mathtt{new}}$ covered by $\mathtt{a}_\mathtt{i}^{\mathtt{new}}$ there exists a $\mathtt{p}_{\mathtt{ij}}^{\mathtt{old}}$ covered by $\mathtt{a}_\mathtt{i}^{\mathtt{old}}$ such that $\mathtt{p}_{\mathtt{ij}}^{\mathtt{new}}=\mathtt{p}_{\mathtt{ij}}^{\mathtt{old}}$,
- for each $\mathtt{q}_{\mathtt{ij}}^{\mathtt{new}}$ covered by $\mathtt{a}_\mathtt{i}^{\mathtt{new}}$ there exists a $\mathtt{q}_{\mathtt{ij}}^{\mathtt{old}}$ covered by $\mathtt{a}_\mathtt{i}^{\mathtt{old}}$ such that $\mathtt{q}_{\mathtt{ij}}^{\mathtt{new}}=\mathtt{q}_{\mathtt{ij}}^{\mathtt{old}}$.

For each dimension interval a_i over $\mathtt{L}_\mathtt{i}^0$ we can construct a corresponding interval b_i over $\mathtt{L}_\mathtt{i}^{\mathtt{old}}$, such that if $a_i=(\mathtt{A}_\mathtt{s},\mathtt{A}_\mathtt{e})$, then $b_i=(\mathtt{B}_\mathtt{s},\mathtt{B}_\mathtt{e})=(\mathtt{anc}_{\mathtt{L}_\mathtt{i}^0}^{\mathtt{L}_\mathtt{i}^{\mathtt{old}}}(\mathtt{A}_\mathtt{s}),\mathtt{anc}_{\mathtt{L}_\mathtt{i}^0}^{\mathtt{L}_\mathtt{i}^{\mathtt{old}}}(\mathtt{A}_\mathtt{e}))$. Algorithm `Cube_Usability` produces exactly these DI's (remember that the domains of all the levels are isomorphic to the integers, which means that if $\mathtt{x}\leq\mathtt{l}$ then `x<succ(l)` --transformation employed for <,> when the edge of a partition is involved). Moreover, if we consider an arbitrary $\mathtt{a}_\mathtt{i}^{\mathtt{new}}$ and the respective $\mathtt{b}_\mathtt{i}^{\mathtt{new}}$ produced by Algorithm `Cube_Usability`, and their respective $\mathtt{a}_\mathtt{i}^{\mathtt{old}}$, $\mathtt{b}_\mathtt{i}^{\mathtt{old}}$ such that $\mathtt{a}_\mathtt{i}^{\mathtt{new}}\subseteq_{\mathbf{L}}\mathtt{a}_\mathtt{i}^{\mathtt{old}}$ then, $\mathtt{b}_\mathtt{i}^{\mathtt{new}}\subseteq_{\mathbf{L}}\mathtt{b}_\mathtt{i}^{\mathtt{old}}$.

Also, the sets of partitions $\mathbf{p}^{\mathrm{new}},\mathbf{q}^{\mathrm{new}}$ covered by $\mathtt{b}_\mathtt{i}^{\mathtt{new}}$ are a subset of the respective partitions $\mathbf{p}^{\mathrm{old}},\mathbf{q}^{\mathrm{old}}$ covered by $\mathtt{b}_\mathtt{i}^{\mathtt{old}}$. This means that if we transform $\varphi'_{\mathtt{new}},\varphi'_{\mathtt{old}}$ at the levels of $\mathtt{c}^{\mathtt{old}}$, as suggested by Algorithm `Cube_Usability`, then for each DI of $\varphi'_{\mathtt{new}},\varphi'_{\mathtt{old}}$ there exists the respective DI at the level of $\mathtt{c}^{\mathtt{old}}$ and $\mathtt{b}_\mathtt{i}^{\mathtt{new}}\subseteq_{\mathbf{L}}\mathtt{b}_\mathtt{i}^{\mathtt{old}}$. We will call the result of the transformation of the expressions $\varphi'_{\mathtt{new}},\varphi'_{\mathtt{old}}$, $\mathtt{y}'_{\mathtt{new}},\mathtt{y}'_{\mathtt{old}}$ respectively. Remember that although $\varphi'_{\mathtt{new}},\varphi'_{\mathtt{old}}$ were defined at the detailed levels, $\mathtt{y}'_{\mathtt{new}},\mathtt{y}'_{\mathtt{old}}$ are defined at the levels of $\mathtt{c}^{\mathtt{old}}$. Obviously, since for all the DI's of $\mathtt{y}'_{\mathtt{new}},\mathtt{y}'_{\mathtt{old}}$, $\mathtt{b}_\mathtt{i}^{\mathtt{new}}\subseteq_{\mathbf{L}}\mathtt{b}_\mathtt{i}^{\mathtt{old}}$, we can claim that $\mathtt{y}'_{\mathtt{new}}\subseteq\mathtt{y}'_{\mathtt{old}}$ and thus $\mathtt{y}'_{\mathtt{new}}\subseteq_{[\mathtt{L}_1^{\mathtt{old}},\ldots,\mathtt{L}_\mathtt{n}^{\mathtt{old}}]}\mathtt{y}'_{\mathtt{old}}$. Using the definition of $\mathbf{L}$-containment, and based on the fact that the partitions covered are identical, we can deduce that for an arbitrary $\mathtt{DS}^0$

$$\sigma_{\mathtt{y}'_{\mathtt{new}}}(\mathtt{c}^{\mathtt{old}})\equiv(\mathtt{DS}^0,\varphi'_{\mathtt{new}}\wedge\varphi'_{\mathtt{old}},[\mathtt{L}_1^{\mathtt{old}},\ldots,\mathtt{L}_\mathtt{n}^{\mathtt{old}},\mathtt{M}_{\mathtt{old}}],\mathtt{agg}_{\mathtt{old}}(\mathtt{M}^0))$$

(although φ'_{new} is a subset of the detailed equivalent of y'_{new} and the reverse does not hold)

Then, all we have to show is that what holds for $\varphi'_{new}, \varphi'_{old}$ holds also for $\varphi_{new}, \varphi_{old}$. Yet, this is trivial, since $\varphi'_{new}, \varphi'_{old}$ are equivalent to $\varphi_{new}, \varphi_{old}$. Consequently, if we transform φ_{old} at the levels of c^{old}, we obtain the expression $\varphi_{n@o}$ and the following hold:

$$\sigma_{\varphi_{n@o}}(c^{old}) \equiv (DS^0, \varphi'_{new} \wedge \varphi'_{old}, [L_1^{old}, \ldots, L_n^{old}, M_{old}], agg_{old}(M^0))$$

$$\equiv (DS^0, \varphi_{new} \wedge \varphi_{old}, [L_1^{old}, \ldots, L_n^{old}, M_{old}], agg_{old}(M^0))$$

$$\equiv (DS^0, \varphi_{new}, [L_1^{old}, \ldots, L_n^{old}, M_{old}], agg_{old}(M^0)).$$

The effect of Lines 19-20 of Algorithm `Cube_Usability` has been covered for **Case 1**. ■