A Personalization Framework for OLAP Queries

Ladjel Bellatreche
LISI / ENSMA
Téléport 2, 1, av. C. Ader
86960 Futuroscope, FRANCE
bellatreche@ensma.fr

Arnaud Giacometti
Patrick Marcel
Hassina Mouloudi
Université François-Rabelais de Tours, LI
3, pl. Jean Jaurès
41000 Blois, FRANCE
giaco@univ-tours.fr

Dominique Laurent
Université de Cergy Pontoise, LICP
95302 Cergy-Pontoise, FRANCE
dominique.laurent@dept-info.u-cergy.fr

ABSTRACT

OLAP users heavily rely on visualization of query answers for their interactive analysis of massive amounts of data. Very often, these answers cannot be visualized entirely and the user has to navigate through them to find relevant facts.

In this paper, we propose a framework for personalizing *OLAP* queries. In this framework, the user is asked to give his (her) preferences and a visualization constraint, that can be for instance the limitations imposed by the device used to display the answer to a query. Given this, for each query, our method computes the part of the answer that respects both the user preferences and the visualization constraint. In addition, a personalized structure for the visualization is proposed.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query Processing*

General Terms

Algorithms, Experimentation

Keywords

Multidimensional Databases, OLAP, Visualization, Personalization, Preferences

1. INTRODUCTION

In the area of data warehousing and *OLAP*, the user is provided with massive amount of data targeted for decision making [5]. Very often, the answer to a query is likely not to fit entirely on the device used for visualization. Thus the decision process is based on an interactive analysis, supported by a query language, which offers primitives to navigate through these data [14]. In such a context, it would be very helpful to provide users with the most relevant visualization of the query answer.

In the context of traditional databases, a technique called query personalization consists in taking user preferences into account when answering a query [12]. Thus different users may obtain different responses to the same query, according to their profiles.

In this paper, we propose a technique that exploits user profiles for displaying the best visualization to the user in an *OLAP* context.

1.1 Motivating example

Suppose that a user has to make a decision by using his (her) favorite *OLAP* system. The decision concerns sales of items of products in different locations, by different salespersons at different periods of time. The sales data are stored in a star schema (see Figure 1).

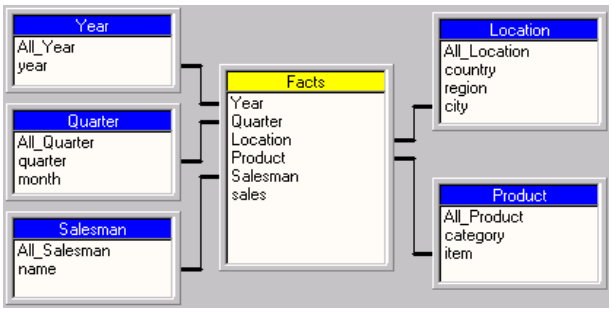

Figure 1: The star schema queried

Suppose that the user wants to know the amount of *sales* detailed by *category* of *products*, *year* and *region*.

Obviously, if the number of categories and/or the number of years are large, the complete answer cannot be visualized entirely on the screen. Moreover, there are many ways to present this result. For example, if the decision maker is responsible for the sales in region *north*, (s)he will be more interested in the visualization presented in Figure 2.

On the other hand, if the user profile indicates that the decision maker is responsible for recent sales of *drink* and *food* whatever the region, then (s)he will be more interested in the visualization presented in Figure 3.

It is important to note that the sets of facts displayed in the two visualizations of Figure 2 and Figure 3 are both subsets of the same answer set. These subsets as well as the way

DOLAP'05, November 4–5, 2005, Bremen, Germany.

region=north

	2002	2003	2004	2005
food	72,00	50,00	33,00	89,00
drink	26,00	20,00	25,00	77,00
cloth	56,00	30,00	32,00	60,00
book	45,00	50,00	32,00	51,00

Figure 2: Visualization 1

		north	south	east	west
2005	drink	77,00	54,00	55,00	33,00
	food	89,00	61,00	30,00	41,00
2004	drink	25,00	50,00	49,00	32,00
	food	33,00	44,00	59,00	27,00

Figure 3: Visualization 2

they are presented, i.e., the structure of each visualization, depend on the user preferences.

1.2 Contribution

In this paper, we propose a method that, given an *OLAP* query, displays the best answer for a user w.r.t. his (her) profile. More precisely:

- We design a framework for expressing the problem of finding the most interesting visualization in an *OLAP* context.
- We propose a definition of user profiles in an *OLAP* context.
- We give an algorithm that finds the most interesting visualization w.r.t. the user profile.

1.3 Related Work

Handling user preferences is an important issue in current information systems, which has motivated many research efforts since many years. Nevertheless, only recently did the Database community pay attention to this research area.

In the context of relational databases, different *models of user preferences* have been proposed. In the *qualitative approach*, user preferences are generally represented by pre-orders or orders between tuples [11, 6], whereas in the *quantitative approach* [1], user preferences are specified using *scoring functions* that associate *degrees of interest* with tuples (a tuple t is preferred to another tuple t' if the score of t is higher than that of t'). Note that the qualitative approach is more general than the quantitative approach, since only total orders can be defined from scoring functions, whereas the qualitative approach can use both total or partial order.

More recently, it has been proposed to express user preferences by degrees of interest associated with values of attributes, or more generally with atomic selection or join conditions [12]. In our framework, user preferences are defined by a total pre-order over the set of members of all dimensions. Therefore, our approach is similar to that in [12], since a total pre-order on members can be defined from degrees of interest (a member or value of an attribute is preferred to another member if its degree of interest is higher). However, we also introduce *visualization constraints* that allow users to specify the limitations imposed by the device to display the answers to their queries. Visualization constraints can also be used to control the form of the visualization of a query result. For example, using visualization constraints, a user can specify that a particular dimension must always appear on a given axis.

The *introduction of user preferences in relational databases* can be done at different levels. In [6], the author proposes a simple *embedding of user preferences formulas into relational algebra*. In this way, the user can specify in every query what are the most interesting tuples of its answer. Then, the tuples of the answer can be computed and ranked based on these preferences. Another approach consists in *storing user profiles* that represent preferences. In this case, when a user submits a query, this query is *personalized* using the preferences stored in his (her) profile [12]. For example, the personalization of a user query can add selection conditions to a query, meaning that the user obtains a subset of the answer to the initial query. In general, it is expected that this subset contains the most interesting tuples (with respect to the user preferences) of the global answer. In our approach, we store in user profiles both preferences that rank members of all dimensions, and visualization constraints that control the form of the visualization of the query results. Moreover, we use these profiles both *to personalize user queries* (in the sense that selection conditions can be added to the original user query) and *to personalize the visualizations of the answers*. Note that in our approach, the personalization of a user query and of its visualization are combined, since we compute the most interesting part of the query answer that can be visualized with respect to the visualization constraints.

1.4 Outline

The next section introduces our approach, first intuitively and then formally. Section 3 presents our algorithm for personalizing visualizations. The consequences of personalization on *OLAP* query optimization are discussed in Section 4. Section 5 deals with the problem of computing interesting visualizations in more general terms and concludes the paper.

2. PROBLEM AND DEFINITIONS

In this section, we first give the intuitions underlying our approach and then, we present the basic definitions.

2.1 Problem

Suppose the user has issued a query and the expected answer is a cube C. In many cases, the large number of facts in C prevents it from being visualized entirely. Thus the user has to navigate through the cube to reach relevant facts. Navigation is based on restructuring operations that change the presentation of the cube on the screen. This presentation is called *structure* in the following, and represents the nesting of dimensions on axes. If the user is provided with a language allowing multidimensional expressions, like for instance Microsoft *MDX* [7], the structure of the cube can be defined in the query itself. Otherwise, the structure is imposed by the system.

In this paper, we deal with the problem of finding the

most interesting sub-cube of a cube C that can be visualized entirely.

2.1.1 Visualization constraints

Given a cube C, we are interested in knowing if C can be visualized entirely or not. For example, a common criterion a cube C must satisfy to be visualized entirely is the following: C corresponds to one slice of the multidimensional data set that can fit on the user's screen. That kind of criterion is called a *visualization constraint* in the following.

Visualization constraints can also be used to define precisely the structure of a cube. However, if the structure of a given cube C is not precisely defined by the visualization constraints, then the problem becomes to find not only the most interesting sub-cube of C that can be visualized, but also a structure for this cube that respects the visualization constraints.

Note that in our approach, we only consider visualization constraints that are anti-monotone, meaning that if a cube C_1 can be visualized and C_2 is a sub-cube of C_1, then C_2 can also be visualized (i.e., if a cube fits on the screen, a part of it also fits on the screen).

2.1.2 User preferences

Obviously not any sub-cube of a cube C is interesting for the user. Interestingness is defined by *user preferences* (or user profile). In our approach, these preferences allow to define a pre-ordering on cubes, so that the problem is to find the most interesting sub-cubes of C w.r.t. this pre-ordering that can be visualized.

There are many ways to model user preferences and to define pre-orderings on cubes. In this paper, user preferences are specified by a total pre-ordering on members of all dimensions of cubes, and the pre-ordering defined over cubes is based on this pre-ordering over members. Moreover, the pre-ordering over cubes is defined in such a way that the following monotonicity property is satisfied: if C_1 is a sub-cube of C_2, then C_1 is less interesting than C_2. This is so because, if C_1 is a sub-cube of C_2, then C_1 contains less information than C_2 and thus, is less interesting than C_2.

2.1.3 Problem

We are looking for the most interesting sub-cubes of a cube C that can be visualized. Denoting by C^* one of these sub-cubes, C^* is such that:

- There exists a structure S^* that allows to visualize it.
- There does not exist a sub-cube of C more interesting than C^* that can be visualized, i.e., C^* is maximal w.r.t. the pre-ordering over cubes.

Note that if visualization constraints do not define precisely the structure of an optimal sub-cube C^*, a structure must be found that respects both user preferences and visualization constraints. This means that our framework personalizes both the set of facts to be presented to the user, and the presentation of these facts.

We now turn to the formal definitions.

2.2 Definitions

In our framework, a cube is simply a set of dimensions and a function that associates combinations of members to measures. In the following, we consider a fixed set $\mathcal{D}$ of dimensions, each dimension $D \in \mathcal{D}$ being a finite set of members. Moreover, we say that a member m belongs to $\mathcal{D}$, denoted $m \in \mathcal{D}$, if there exists D in $\mathcal{D}$ such that $m \in D$.

Definition 1. **- Cube.** *An N-dimensional cube C is a tuple $C = \langle D_1, \ldots, D_N, f\rangle$ where:*

- *$\forall i \in [1, N], D_i$ is a dimension in $\mathcal{D}$,*
- *f is a mapping from $D_1 \times \ldots \times D_N$ to $\mathbb{R}$.*

Given two N-dimensional cubes $C' = \langle D'_1, \ldots, D'_N, f'\rangle$ and $C = \langle D_1, \ldots, D_N, f\rangle$, C' is a sub-cube of C, noted $C' \subseteq C$, if $\forall i \in [1, N], D'_i \subseteq D_i$ and f' is the restriction of f to $D'_1 \times \ldots \times D'_N$, noted $f_{|D'_1 \times \cdots \times D'_N}$. We also use the following notations:

- $\mathcal{C}$ is the set of all cubes,
- $M(C)$ is the set of all members of C, i.e., $M(C) = \bigcup_{i \in [1,N]} D_i$.

A structure S of a cube C specifies how the facts of C are presented to the user.

Definition 2. **- Structure.** *A K-dimensional structure S of an N-dimensional cube $C = \langle D_1, \ldots, D_N, f\rangle$ is a K-tuple $S = \langle S_1, \ldots, S_K\rangle$ where $K \leq N$ and S_k ($k = 1, \ldots, K$) are disjoint subsets of $\{D_1, \ldots, D_N\}$. We denote by $\mathcal{S}$ the set of all structures.*

Example 1. *In the context of our motivating example, let $C = \langle D_1, D_2, D_3, f\rangle$ be a 3-dimensional cube with $D_1 = \{north, south, east, west\}$, $D_2 = \{food, drink, cloth, book\}$ and $D_3 = \{2000, 2001, 2002, 2003, 2004, 2005\}$. In Figure 2, we can see that $f(north, food, 2002) = 72,00$.*

On the other hand, let $S_1 = \langle\{D_2\}, \{D_3\}\rangle$ and $S_2 = \langle\{D_3, D_2\}, \{D_1\}\rangle$. S_1 is the structure of the sub-cube of C shown in Figure 2, and S_2 is the structure of the sub-cube of C shown in Figure 3. Note that S_1 and S_2 are two 2-dimensional structures. Moreover, we can see that the dimension D_1 does not belong to S_1, meaning that this dimension is not placed on a visible axis of the visualization. Thus, in Figure 2, we can only see the total sales for one member of this dimension, namely, the north region.

Definition 3. **- Visualization.** *A visualization is a tuple $\langle C, S\rangle$ where $C \in \mathcal{C}$ is a cube and $S \in \mathcal{S}$ is a structure of C.*

Given a cube and a structure, we define a visualization constraint as a function indicating whether the cube can be visualized using this structure.

Definition 4. **- Visualization constraint.** *A visualization constraint v is a boolean function defined over $\mathcal{C} \times \mathcal{S}$. Given a cube $C \in \mathcal{C}$ and a structure $S \in \mathcal{S}$, we say that C can be visualized with respect to v if $v(C, S) = true$.*

Moreover, a visualization constraint is anti-monotone *if for every pair of cubes $(C, C') \in \mathcal{C}^2$, if $C \subseteq C'$ and there exists a structure $S' \in \mathcal{S}$ such that $v(C', S') = true$, then there exists a structure $S \in \mathcal{S}$ such that $v(C, S) = true$.*

In the following, $\mathcal{V}$ denotes the set of all anti-monotone visualization constraints.

EXAMPLE 2. *Suppose a cube must be displayed as a bi-dimensional cross-tab, i.e., with 2 axes. Suppose also that each axis admits a maximal number G of positions (in Figures 2 and 3, we have $G = 4$). We can define a visualization constraint v for every cube $C = \langle D_1, \ldots, D_N, f\rangle$ and structure $S = \langle S_1, S_2\rangle$ by $v(C, S) = true$ if:*

1. $\forall k \in [1, 2], \prod_{D_i \in S_k} |D_i| \leq G$.
2. $\forall i \in [1, N]$, *if* $D_i \notin (S_1 \cup S_2)$, *then* $|D_i| = 1$.

In this example, a cube C can be visualized w.r.t. v if:

1. *for each axis, the product of the cardinalities of the dimensions on this axis is less than or equal to G, and*
2. *there is only one member in every dimension that does not appear on the visible axes.*

Moreover, it is easy to see that v is anti-monotone because if $v(C, S) = true$, then for every C' such that $C' \subseteq C$, we have that $v(C', S) = true$.

In our approach, we consider that a pre-ordering $\preceq_P$ over $\mathcal{C}$ is defined by user preferences. Given two cubes C and C', we say that C is *less interesting* than C' if $C \preceq_P C'$.

The pre-ordering $\preceq_P$ induces an equivalence relation over $\mathcal{C}$, denoted by $\equiv$, defined as follows: given C and C' in $\mathcal{C}$, $C \equiv C'$ if $C \preceq_P C'$ and $C' \preceq_P C$ hold. Intuitively, two cubes are equivalent modulo $\equiv$ if they are "equally interesting".

It is important to note that, in general if $C \subseteq C'$, then we have $C \preceq_P C'$, since C' contains more information than C. On the other hand, if $C \subseteq C'$ and C can be visualized w.r.t. a visualization constraint $v \in \mathcal{V}$, this does not necessarily imply that C' can be visualized w.r.t. v. This is so because visualization constraints are anti-monotone, but not monotone.

Based on these definitions, given a cube C, a pre-ordering over $\mathcal{C}$ and a visualisation constraint $v \in \mathcal{V}$, the problem is to find the most interesting sub-cubes of C that can be visualized w.r.t. v.

DEFINITION 5. **- Problem.** *Given a cube C, a visualization constraint v in $\mathcal{V}$ and a pre-ordering $\preceq_P$ over $\mathcal{C}$, the problem is to find a visualization $\langle C^*, S^*\rangle$ of C such that $C^* \in max_{\preceq_P}\{C' \subseteq C \mid \exists S', v(C', S') = true\}$ and $v(C^*, S^*) = true$.*

Note that several visualizations can be solution of the same problem since $\preceq_P$ is a pre-ordering over $\mathcal{C}$. Moreover, if $\preceq_P$ is a *total* pre-ordering, then all the sub-cubes C^* of the optimal visualizations $\langle C^*, S^*\rangle$ are equivalent modulo $\equiv$.

2.3 Complexity

In general, the search problem presented in Definition 5 is NP-hard. Indeed, we can show that a subclass of this general problem is similar to the NP-hard Knapsack Problem [9].

Assume that a measure of utility $\mu(m) \in \mathbb{N}$ is associated with every member m of all dimensions. We define:

- The function F_μ for every cube $C \in \mathcal{C}$ by: $F_\mu(C) = \sum_{m \in M(C)} \mu(m)$.
- The order $\preceq_\mu$ over $\mathcal{C}$ by: $C \preceq_\mu C'$ iff $F_\mu(C) \leq F_\mu(C')$.

It is easy to see that, for every pair of cubes $(C, C') \in \mathcal{C}^2$, if $C \subseteq C'$, then we have $C \preceq_\mu C'$.

On the other hand, given a structure $S = \langle S_1, \ldots, S_K\rangle$ of a cube C and an integer G, we define the visualization constraint v_G for every cube $C \in \mathcal{C}$ by: $v_G(C, S) = true$ if $(\prod_{D \in S_k} |D| \leq G)$, for every $k \in [1, K]$.

Given a one-dimensional cube $C = \langle D_1, f\rangle$, our problem is to find a visualization $\langle C^*, S^*\rangle$ such that $C^* \in max_{\preceq_\mu}\{C' \subseteq C \mid (\exists S' \in \mathcal{S})(v_G(C', S') = true)\}$ and $v(C^*, S^*) = true$. A sub-cube C^* is defined by a subset D_1^* of D_1 such that:

1. $F_\mu(C^*) = \sum_{m \in D_1^*} \mu(m)$ is maximal.
2. There exists a structure S^* such that $v_G(C^*, S^*) = true$, that is $|D_1^*| \leq G$ (note that with a 1-dimensional cube C, we can only consider 1-dimensional structures S, i.e., $K = 1$).

Thus, we are looking for a subset D_1^* of D_1 such that its cardinality is less than G and the objective function $\sum_{m \in D_1^*} \mu(m)$ is maximal. This problem corresponds exactly to the Knapsack Problem, which shows that a subclass of our general problem is NP-hard.

3. ALGORITHM

In Section 2.3, we have shown that, in general, the search problem presented in Definition 5 is NP-hard. In this section, we present a particular pre-ordering over $\mathcal{C}$ and a particular set of visualization constraints for which the complexity is polynomial in time. In this setting, we also propose algorithms that:

- Find the most interesting sub-cubes of a cube that can be visualized (see function $PersoVisu$ presented in Figure 4).
- Test if a cube can be visualized w.r.t. a visualization constraint (see function $FindStruct$ presented in Figure 5).

3.1 User Profiles

In our framework, a user profile is defined by user preferences (represented by a total pre-ordering on members) and a visualization constraint.

Given a fixed set $\mathcal{D}$ of dimensions, let $\leq_P$ be a total pre-ordering on members in $\mathcal{D}$. For every pair of members $(m, m') \in \mathcal{D}^2$, we say that m' *is preferred to* m if $m \leq_P m'$. Let $\sim_P$ be the equivalence relation defined for every pair of members $(m, m') \in \mathcal{D}^2$ by $m \sim_P m'$ if $m \leq_P m'$ and $m' \leq_P m$ hold, meaning that m and m' are equally preferred.

We now introduce a total pre-ordering over $\mathcal{C}$, based on the *lectic order* used in formal concept analysis [8]. Intuitively, given two cubes C_1 and C_2, C_1 is less interesting than C_2, noted $C_1 \preceq_P C_2$, if for every member m_1 in C_1 but not in C_2, there exists a member m_2 in C_2 but not in C_1 such that m_2 is preferred to m_1. Formally:

DEFINITION 6. **- User Preferences.** *Let C_1 and C_2 be two cubes of $\mathcal{C}$. Given a total pre-ordering $\leq_P$ on members, C_1 is less interesting than C_2, denoted by $C_1 \preceq_P C_2$, if for every $m_1 \in M(C_1) \setminus M(C_2)$, there exists $m_2 \in M(C_2) \setminus M(C_1)$ such that $m_1 \leq_P m_2$.*

Note that if $C_1 \subseteq C_2$, then we have $M(C_1) \setminus M(C_2) = \emptyset$, and thus, $C_1 \preceq_P C_2$. More generally, the following proposition states that two cubes are always comparable w.r.t. $\preceq_P$, based only on the the preferred members that distinguish C_1 from C_2 and C_2 from C_1, i.e., the members in $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2)))$.

PROPOSITION 1. *The relation $\preceq_P$ defined over $\mathcal{C}$ is a total pre-ordering. Moreover, for all cubes C_1 and C_2 in $\mathcal{C}$, we have: $C_1 \preceq_P C_2$ if and only if $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2))) \cap C_2 \neq \emptyset$.*

PROOF: See Appendix.

We note that Proposition 1 above implies that two cubes C_1 and C_2 are equivalent modulo $\equiv$ (i.e., $C_1 \preceq_P C_2$ and $C_2 \preceq_P C_1$) if and only if the set of preferred members that distinguish them contains members of C_1 and members of C_2.

EXAMPLE 3. *Let C be the cube defined in Example 1. Let C_1 and C_2 be the sub-cubes of C defined by:*

- $C_1 = \langle D_{11}, D_{12}, D_{13}, f_1 \rangle$ *with* $D_{11} = \{north\}$, $D_{12} = \{food, drink, cloth, book\}$, $D_{13} = \{2002, 2003, 2004, 2005\}$ *and* $f_1 = f_{|D_{11} \times D_{12} \times D_{13}}$.
- $C_2 = \langle D_{21}, D_{22}, D_{23}, f_2 \rangle$ *with* $D_{21} = \{north, south, east, west\}$, $D_{22} = \{food, drink\}$, $D_{23} = \{2004, 2005\}$ *and* $f_2 = f_{|D_{21} \times D_{22} \times D_{23}}$.

We have $M(C_1) = \{north, food, drink, cloth, book, 2002, 2003, 2004, 2005\}$ and $M(C_2) = \{north, south, east, west, food, drink, 2004, 2005\}$. Let $\leq_P$ be the total pre-ordering defined over $D_1 \cup D_2 \cup D_3$ by: $(drink \sim_P food) >_P 2005 >_P 2004 >_P (north \sim_P south \sim_P east \sim_P west) >_P (2000 \sim_P 2001 \sim_P 2002 \sim_P 2003) >_P (cloth \sim_P book)$. This total pre-ordering defines a user's profile that leads to the visualization presented in Figure 3. On the other hand, we have:

- $M(C_1) \setminus M(C_2) = \{cloth, book, 2002, 2003\}$,
- $M(C_2) \setminus M(C_1) = \{south, east, west\}$.

Thus, the most interesting members w.r.t. $\leq_P$ that distinguish C_1 from C_2 are south, east and west. Since all these members belong to C_2 but not to C_1, Proposition 1 shows that $C_1 \preceq_P C_2$.

We now define the type of visualization constraints that we consider. Let $T = \langle T_1, \ldots, T_K \rangle$ be a K-dimensional structure and $G = \langle G_1, \ldots, G_K \rangle$ be a K-tuple of integers. For every cube $C = \langle D_1, \ldots, D_N, f \rangle$ and structure $S = \langle S_1, \ldots, S_L \rangle$, we define the visualization constraint $v_{T,G}$ by: $v_{T,G}(C, S) = true$ if $L = K$ and

- For $k \in [1, K]$, we have $T_k \subseteq S_k$
  This constraint means that the user wants to see the dimensions in T_k on axis k.
- For $k \in [1, K]$, we have $\prod_{D_i \in S_k} |D_i| \leq G_k$
  This constraint means that the user can see a maximal number G_k of positions on axis k.
- For $i \in [1, N]$, if $D_i \notin (S_1 \cup \cdots \cup S_K)$, then $|D_i| = 1$
  This constraint means that every dimension that does not appear on the visible axes must contain only one member.

In the following, we denote by $\mathcal{V}^*$ the set of all visualization constraints of the form $v_{T,G}$. It is easy to see that every visualization constraint in $\mathcal{V}^*$ is anti-monotone as defined in Definition 4.

3.2 Computing Personalized Visualizations

In this section, we present an algorithm called $PersoVisu$ that computes a personalized visualization of a cube C w.r.t. to a visualization constraint $v \in \mathcal{V}^*$ and a total pre-ordering $\preceq_P$ over $\mathcal{C}$ defined by user preferences. Given a cube C, this algorithm, presented in Figure 4, computes one of the most interesting sub-cube C^* of C that can be visualized. Note that C^* is a representative of the equivalence class modulo $\equiv$ of optimal sub-cubes of C. Moreover, every sub-cube in this equivalence class can be obtained by substituting members of C^* with equivalent members of C not in C^*.

First (see steps 1 and 2), $PersoVisu$ computes one of the most interesting members $m_i \in max_{\leq_P}(D_i)$ for every dimension D_i of C, and initializes every set D_i^* to the singleton $\{m_i\}$ $(i = 1, \ldots, N)$. Note that the initial sets D_i^* define a sub-cube $C^* = \langle D_1^*, \ldots, D_N^*, f_{|D_1^* \times \cdots \times D_N^*} \rangle$ of C that contains only one cell. In our algorithm, we assume that a cube with only one cell can always be visualized, i.e., the maximal number of positions on every axis is strictly greater than 0.

Then, $PersoVisu$ adds new members m^* to the sets D_i^* so that C^* is maximal w.r.t. to $\preceq_P$ and the visualization constraint v can be satisfied. At the beginning of the main loop (steps 4-12), the set $\mathcal{M}$ contains the members of C not in $\bigcup_{i=1}^{N} D_i^*$. Then:

- At step 5, one of the most interesting members of $\mathcal{M}$ is selected.
- At steps 6 and 7, the dimension D_i that contains m^* is found and m^* is added to D_i^*.
- At step 9, it is checked whether there exists a structure S^* such that $v(C^*, S^*) = true$ where C^* is the sub-cube of C characterized by the dimensions D_i^*. If such a structure does not exist, then m^* is removed from D_i^*. Moreover, all members of $\mathcal{M}$ that belong to the dimension m^* belongs to are removed from $\mathcal{M}$. Indeed, if the visualization constraint cannot be satisfied when m^* is added to D_i^*, the same happens for every member of the dimension m^* belongs to.

When $\mathcal{M}$ is empty, no members can be added to the sets D_i^* so that the visualization constraint can be satisfied. Thus, the main loop is finished. At step 14, the algorithm finally computes a structure S^* of C^* such that $v(C^*, S^*) = true$. By construction of the sets D_i^*, it is easy to see that such a structure exists.

Note that, in this algorithm, we do not explicit how it is checked whether there exists a structure S^* of C^* such that $v(C^*, S^*) = true$. In Section 3.3, we show how this test can be performed for visualization constraints in $\mathcal{V}^*$. Then, in Section 3.4, we analyse the complexity of our algorithm.

THEOREM 1. *Let C be a cube in $\mathcal{C}$. Given a visualization constraint in $\mathcal{V}^*$ and a total pre-ordering $\leq_P$ on members, the algorithm $PersoVisu$ computes a maximal interesting sub-cube of C that can be visualized.*

PROOF: See Appendix.

Function *PersoVisu*

Input: A cube $C = \langle D_1, \ldots, D_N, f\rangle$
A visualization constraint $v \in \mathcal{V}^*$
A total pre-ordering $\leq_P$ on members of all dimensions
Output: A personalized visualization $\langle C^*, S^*\rangle$ of C

1. **For** $i = 1$ to N
2. **Let** $D_i^* = \{m_i\}$ where $m_i \in max_{\leq_P}(D_i)$
3. **Let** $\mathcal{M} = \bigcup_{i=1}^{N}(D_i \setminus D_i^*)$
4. **While** $(\mathcal{M} \neq \emptyset)$ **do**
5. **Let** $m^* \in max_{\leq_P}(\mathcal{M})$
6. **Let** $i \in \{1, \ldots, N\}$ such that $m^* \in D_i$
7. **Let** $D_i^* = D_i^* \cup \{m^*\}$
8. **Let** $\mathcal{C}^* = \langle D_1^*, \ldots, D_N^*, f^*\rangle$ where $f^* = m_{|D_1^* \times \cdots \times D_N^*}$
9. **If** $(\nexists S^* \in \mathcal{S})(v(\mathcal{C}^*, S^*) = true)$ **then**
10. $D_i^* = D_i^* \setminus \{m^*\}$ and $\mathcal{M} = \mathcal{M} \setminus D_i$
11. **Else** $\mathcal{M} = \mathcal{M} \setminus \{m^*\}$
12. **end while**
13. **Let** $\mathcal{C}^* = \langle D_1^*, \ldots, D_N^*, f^*\rangle$
14. **Let** $S^* \in \mathcal{S}$ such that $v(\mathcal{C}^*, S^*) = true$
15. **Return** $\langle C^*, S^*\rangle$

Figure 4: Computation of a personalized visualization

Example 4. *Let C be the cube defined in Example 1 and $\preceq_P$ the pre-ordering on members defined in Example 3. Consider the visualization constraint $v_{T,G} \in \mathcal{V}^*$ defined by $T = \langle \emptyset, \emptyset\rangle$ and $G = \langle 4, 4\rangle$. In this case, the user does not precise on which axes the dimensions of C are to be displayed, and states that no more than 4 rows and 4 columns can be seen horizontally and vertically.*

PersoVisu computes first the sets $D_1^* = \{north\}$, $D_2^* = \{drink\}$ and $D_3^* = \{2005\}$, and then, initializes the set $\mathcal{M}$ to $(D_1 \cup D_2 \cup D_3) \setminus \{north, drink, 2005\}$. In its main loop, PersoVisu successively inserts food to D_2^*, 2004 to D_3^*, south to D_1^*, east to D_1^* and west to D_1^*. At this stage, we have:

- $D_1^* = \{north, south, east, west\}$, $D_2^* = \{drink, food\}$, $D_3^* = \{2004, 2005\}$, *and*
- $\mathcal{M} = \{cloth, book, 2000, 2001, 2002, 2003\}$.

Moreover, the visualization constraint $v_{T,G}$ is satisfied if D_2^* and D_3^* are nested on the same axis, and D_1^* is placed alone on a second axis.

Then, when PersoVisu tries to insert cloth into D_2^*, no structure S^* satisfying the visualization constraints $v_{T,G}$ can be found. Therefore, PersoVisu removes cloth and book from $\mathcal{M}$. At the next iteration, for the same reason, the years 2000 to 2003 are removed from $\mathcal{M}$. Thus, $\mathcal{M}$ is empty and the main loop is finished. PersoVisu terminates by computing a structure S^* such that $v_{T,G}(C^*, S^*) = true$. For example, $S^* = \langle\{D_2^*, D_3^*\}, \{D_1^*\}\rangle$, as presented in Figure 3, can be output.

3.3 Satisfying Visualization Constraints

Given a cube C and a visualization constraint v in $\mathcal{V}^*$, we show in this section how to test if v can be satisfied, i.e., if there exists a structure S such that $v(C, S) = true$. The following proposition shows that, in general, this problem is NP-complete.

Proposition 2. *Let C be a cube in $\mathcal{C}$ and v be a visualization constraint in $\mathcal{V}^*$. The problem to test if there exists a structure S of C such that $v(C, S) = true$ is NP-complete.*

Proof: See Appendix.

In what follows, we show that when $K = 2$, it is possible to solve this problem with an algorithm in $O(N\, G_m^2)$, where N is the number of dimensions of C and G_m is the maximal number of positions that can be visualized on the axes.

Let $v_{T,G}$ be a visualization constraint in $\mathcal{V}^*$ with $T = \langle T_1, T_2\rangle$ and $G = \langle G_1, G_2\rangle$. In Figure 5, given a cube $C \in \mathcal{C}$, we propose an algorithm, based on dynamic programming and called *FindStruct*, that tests if there exists a structure S of C such that $v_{T,G}(C, S) = true$. First, this algorithm computes the set $\mathcal{D}' = \{D_{i_1}, \ldots, D_{i_M}\}$ of dimensions that have to be placed on visible axes. Then, it can be seen that, at the m-th iteration $(1 \leq m \leq M)$ of the main loop (see steps 6-16), the cell $t'[k][j]$ of array t' is different from *null* if there exists a partition of $E_m = \{D_{i_1}, \ldots, D_{i_m}\} \subseteq \mathcal{D}'$ into two sets S_1 and S_2 such that, for $p = 1, 2$, $T_p \subseteq S_p$ and $\prod_{D_j \in S_p} |D_j| \leq G_p$. Moreover, if $t'[k][j] \neq null$, the function *FindStruct* uses $t'[k][j]$ to represent one of the partitions $\{S_1, S_2\}$ of E_m that satisfy the constraint, i.e., $S_1 = t'[k][j]$ and $S_2 = E_m \setminus S_1$.

Example 5. *In the context of Example 4, assume that we have a cube $C^* = \langle D_1^*, D_2^*, D_3^*, f^*\rangle$ with $D_1^* = \{north, south, east, west\}$, $D_2^* = \{drink, food\}$, and $D_3^* = \{2004, 2005\}$. For $T = \langle \emptyset, \emptyset\rangle$ and $G = \langle 4, 4\rangle$, we show step by step how the function FindStruct tests if there exists a structure S^* such that $v_{T,G}(C^*, S^*) = true$*

We have first $K_1 = 4$ and $K_2 = K_3 = 2$. Moreover, since $T_1 = T_2 = \emptyset$, we have $G_1' = G_2' = 1$ and $t[1][1] = \emptyset$, the other cells of t being initialized to null. Then, $\mathcal{D}'$ is set to $\{D_1^*, D_2^*, D_3^*\}$.

At the first iteration of the main loop, let $D_i = D_1^*$. Since $t[1][1] = \emptyset$ and $1 * K_1 = 4 \leq 4$, we obtain $t'[4][1] = \{D_1^*\}$ and $t'[1][4] = \emptyset$.

At the second iteration, let $D_i = D_2^*$. Since $t[4][1] = \{D_1^*\}$ and $1 * K_2 \leq 4$, we obtain $t'[4][2] = \{D_1^*\}$. Since $t[1][4] = \emptyset$ and $1 * K_2 \leq 4$, we obtain $t'[2][4] = \{D_2^*\}$.

At the third and last iteration, let $D_i = D_3^*$. Since $t[4][2] = \{D_1^*\}$ and $2 * K_3 \leq 4$, we obtain $t'[4][4] = \{D_1^*\}$. Since $t[2][4] = \{D_2^*\}$ and $2 * K_3 \leq 4$, we obtain $t'[4][4] = \{D_2^*, D_3^*\}$.

At the end, assume that $t[4][4] = \{D_2^*, D_3^*\}$ (this solution is the last stored). In that case, FindStruct returns the structure $\langle\{D_2^*, D_3^*\}, \{D_1^*\}\rangle$, meaning that the visualization constraint $v_{T,G}$ can be satisfied.

3.4 Complexity

In this section, we analyse the time complexity of functions *FindStruct* and *PersoVisu* when 2-dimensional visualizations are considered. Let N be the total number of dimensions of the cube C to be visualized and G_m be the maximal number of cells that can be visualized horizontally or vertically, i.e., $G_m = max_{\leq}\{G_1, G_2\}$.

In the worst case, the set $\mathcal{D}'$ initialized at step 5 in function *FindStruct* contains N dimensions. Thus, N is the maximal number of iterations of the main loop (steps 6-16)

Function *FindStruct*

Input: A cube $C = \langle D_1, \ldots, D_N, f \rangle$
A visualization constraint $v_{T,G} \in \mathcal{V}^*$ with $T = \langle T_1, T_2 \rangle$ and $G = \langle G_1, G_2 \rangle$
Output: A structure $S \in \mathcal{S}$ such that $v(C, S) =$ *true* or *null* if no solution exists

1. **For** $i = 1$ to N **Let** $K_i = |D_i|$
2. **For** every $j \in [1, G_1]$ and $k \in [1, G_2]$
   Let $t[j][k] = null$
3. **For** $k = 1, 2$
   If $T_k = \emptyset$ **then** $G'_k = 1$ **else** $G'_k = \prod_{D_i \in T_k} K_i$
4. **If** $(G'_1 \leq G_1)$ and $(G'_2 \leq G_2)$
   then $t[G'_1][G'_2] = T_1$
   else Return *null*
5. **Let** $\mathcal{D}' = \{D_i \mid i \in [1, N], K_i > 1\} \setminus (T_1 \cup T_2)$
6. **While** $(\mathcal{D}' \neq \emptyset)$ **do**
7. **Let** $D_i \in \mathcal{D}'$ and $\mathcal{D}' = \mathcal{D}' \setminus \{D_i\}$
8. **For** every $j \in [1, G_1]$ and $k \in [1, G_2]$
   Let $t'[j][k] = null$
9. **For** every $j \in [1, G_1]$ and $k \in [1, G_2]$ **do**
10. **If** $(t[j][k] \neq null)$ **do**
11. **If** $(j * K_i \leq G_1)$ **then**
    $t'[j * K_i][k] = t[j][k] \cup \{D_i\}$
12. **If** $(k * K_i \leq G_2)$ **then**
    $t'[j][k * K_i] = t[j][k]$
13. **end if**
14. **end for**
15. **Let** $t = t'$
16. **end while**
17. **For** every $j \in [1, G_1]$ and $k \in [1, G_2]$ **do**
18. **If** $(t[j][k] \neq null)$ **do**
19. **Let** $S_1 = t[j][k]$ and $S_2 = T_2 \cup (\mathcal{D}' \setminus T_1)$
20. **Return** $\langle S_1, S_2 \rangle$
21. **end for**
22. **Return** *null*

Figure 5: Computation of a structure satisfying a visualisation constraint

of *FindStruct*. For every iteration of this main loop, we test if $t[j][k]$ is different than *null*. In that case, we test if we can add the dimension D_i to one of the two axes of the structure. Note that these operations are done for every $k \in [1, G_1]$ and $j \in [1, G_2]$. Thus, the time complexity of every iteration is $O(G_1 \times G_2) = O(G_m^2)$. Hence, since the array t is scanned once more after the main loop, the time complexity of *FindStruct* is $O((N+1).G_m^2) = O(N.G_m^2)$.

We now evaluate how many times the function *FindStruct* is called when a personalized visualization is computed using function *PersoVisu*. The main loop of *PersoVisu* terminates when every dimension D_i of the cube has been removed from $\mathcal{M}$. Thus, the main loop contains at least N iterations. On the other hand, we have to evaluate the maximal number of members m^* that can be added to the sets D_i^*. It is easy to see that the number of members m^* considered is maximal if no dimensions are nested on axes. In that case, each axis k can contain G_k members $(k = 1, 2)$. Thus, the maximal number of members m^* that can be added to the sets D_i^* is $(G_1 + G_2)$. It follows that in the worst case, the main loop of function *PersoVisu* contains $G_1 + G_2 + N$ iterations. Since *FindStruct* is called for every iteration and since this function is called once more to build the structure S^* of the personalized visualization (if any), the time complexity of *PersoVisu* is $O((G_1 + G_2 + N + 1)\ N\ G_m^2) = O(N\ G_m^3 + N^2\ G_m^2)$ in the worst case.

Considering this complexity in time, it is important to note that if the user profiles and the dimensions of the cubes are stored in main memory, then the computation of a personalized visualization can be done in main memory, without accessing the fact table (represented in our framework by a function). Moreover, if the cube C to be visualized is the result of a query q on a cube C_0, i.e., $C = q(C_0)$, it is important to note that the sets D_i^* computed by function *PersoVisu* can be used to add selection conditions to the query q. We are currently investigating [3] how these additional selection conditions can be used to optimize the computation of q by computing *only* the most interesting sub-cube of C that is visualized. The consequences of personalization on *OLAP* query optimization are discussed in the next section.

4. CONSEQUENCES OF QUERY PERSONALIZATION ON OPTIMIZATION

The main consequence of considering user profiles to personalize *OLAP* queries is the addition of new selection and join conditions in the initial queries (see Figure 2, where a selection condition region = "North" has been added). To optimize *OLAP* queries, several techniques were proposed, among which we cite materialized views, indexes, and data partitioning. Note that most algorithms selecting materialized views, indexes, and partitioning schemas consider a set of workloads consisting of *SQL* statements on a database. In this section, we show the impact of our approach on different selection algorithms: materialized view selection problem (*VSP*) [10], index selection problem (*ISP*) [4], and data partitioning selection problem (*PSP*) [2].

4.1 Materialized Views Selection Problem

The *VSP* consists in choosing a set of materialized views defined over a database schema [10], such that the cost of evaluating a set of workload queries is minimized and such

that the selected views fit into a pre-specified storage constraint. Several algorithms were proposed considering SQL queries which include select, project, join, and aggregation operations. These algorithms consider a global query access plan, in which the local access plans for individual queries are merged based on the shared operations (join, union, intersection, etc.) on common data sets (this is called Multiple View Processing Plan ($MVPP$)). Each intermediate node represents a potential view.

In this case, considering user profiles adds new join operations. Thus,the selection of materialized views is more challenging, because the number of intermediate nodes is increased, w.r.t. the initial $MVPP$.

4.2 Index Selection Problem

The task of index selection is, for a given database and a given workload, to select an appropriate set of indexes respecting a storage constraint. A number of indexing strategies have been suggested for data warehouses: bitmap index, B-tree, join Index, and star join index [15].

Considering user profiles, the presence of new selection and join conditions change the input of index selection algorithms and increase the number of sets of candidate indexes for the modified workload. For example, the data warehouse administrator could select indexes like B-trees or bitmap on user profile attributes used in selection conditions, and bitmap join indexes to speed up the new join operations.

4.3 Data Partitioning Selection Problem

The technique of data partitioning consists of decomposing a relational schema of a data warehouse (star schema or snow flake schema) into several sub-schemas, obtained as selections on the initial schema. For example, using the attribute *year* of the dimension table $Year$, the star schema of Figure 1 can be partitioned table in four sub-schemas, where each one represents the sales of the years 2002, 2003, 2004 and 2005, respectively.

Therefore, when considering user profiles, new selection conditions should be taken into account during the fragmentation process, because the complexity of most data partitioning algorithms is proportional to the number of selection predicates [16, 2].

Regarding implementation issues of our approach, an architectural overview of the optimization process is shown in Figure 6. The input is a database and a workload consisting of a set of most frequently SQL queries and their frequencies. When considering user profiles, initial workloads can be rewritten by adding new join and selection conditions to the queries in the initial workload. Then, the data warehouse administrator can use any selection algorithm to pick a set of materialized views, indexes, and a data partitioning schema.

5. CONCLUSION

In this paper, we have proposed an approach for personalizing answers to $OLAP$ queries over cubes. More precisely, we have shown how to compute the most interesting subset of facts in the answer to an $OLAP$ query, as well as a visualization for it. Both the subset of facts and the visualization are computed w.r.t. the user profiles, expressed as a pre-ordering over members of cubes and as a visualization constraint. We also considered the consequences of personalization on $OLAP$ query optimization.

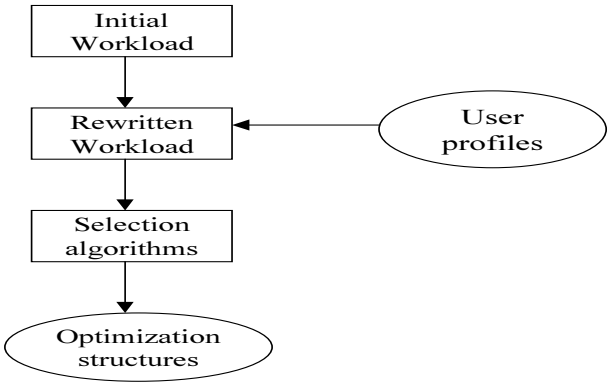

Figure 6: Architecture of the Optimization Process

Based on this work, we are investigating a more general formulation of the problem considered in this paper, namely: Given a cube C, a pre-ordering on $\preceq_P$ defined by user preferences, a visualisation constraint v, and a query language Q over cubes, compute the most interesting cubes such that there exists a structure which allows to visualize them.

In this paper, this problem is restricted to:

- A very simple model of cube, where each dimension consists of only one attribute.
- The operation of selection for the language Q.
- A total pre-ordering on members for user preferences.

We are currently considering the following extensions:

- Enhancing the cube model, e.g., so as to take hierarchically structured dimensions into account.
- Considering a richer language Q, allowing for instance to express grouping and aggregations.
- Defining user preferences based on a pre-ordering over facts, i.e., over members and measures.

Finally, it would be interesting to consider a more sophisticated structure model for cubes, such as the Cube Presentation Model proposed in [13].

6. REFERENCES

[1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 297–306. ACM, 2000.

[2] L. Bellatreche and K. Boukhalfa. An evolutionary approach to schema partitioning selection in a data warehouse environment. *To appear in Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2005)*, pages 115–125, 2005.

[3] L. Bellatreche, A. Giacometti, D. Laurent, P. Marcel, and H. Mouloudi. A framework for combining rule-based and cost-based approaches to optimize OLAP queries. *Numéro spécial, Entrepôts de Données et Analyse en ligne, RNTI, to be published.*, 2005.

[4] S. Chaudhuri. Index selection for databases: A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1313–1323, November 2004.

[5] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *Sigmod Record*, 26(1):65–74, March 1997.

[6] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[7] M. Corporation. OLEDB for OLAP. Available at http://www.microsoft.com/ data/oledb/olap, 1998.

[8] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997. Translator-C. Franzke.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[10] H. Gupta, V. Harinarayan, A. Rajaraman, and J. Ullman. Index selection for olap. *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 208–219, 1997.

[11] W. Kießling. Foundations of preferences in database systems. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 311–322. Morgan Kaufmann, 2002.

[12] G. Koutrika and Y. E. Ioannidis. Personalization of queries in database systems. In *ICDE*, pages 597–608. IEEE Computer Society, 2004.

[13] A. S. Maniatis, P. Vassiliadis, S. Skiadopoulos, and Y. Vassiliou. Advanced visualization for olap. In *DOLAP '03: Proceedings of the 6th ACM international workshop on Data warehousing and OLAP*, pages 9–16, New York, NY, USA, 2003. ACM Press.

[14] P. Marcel. Modeling and querying multidimensional databases: An overview. *Networking and Information Systems Journal*, 2(5-6):515–548, 1999.

[15] P. E. O'Neil and D. Quass. Improved query performance with variant indexes. In J. Peckham, editor, *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 38–49. ACM Press, 1997.

[16] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems : Second Edition*. Prentice Hall, 1999.

APPENDIX

A. PROOFS OF PROPOSITIONS

In this appendix, we give proofs of propositions and theorem.

A.1 Proof of Proposition 1

It is easy to see that $\preceq_P$ is reflexive. Let C_1 and C_2 be two cubes in $\mathcal{C}$. If $C_i \subseteq C_j$, we have $C_i \preceq_P C_j$. If $C_1 \not\subset C_2$ and $C_2 \not\subset C_1$, let $m_1 \in max_{\leq_P}(M(C_1) \setminus M(C_2)$ and $m_2 \in max_{\leq_P}(M(C_2) \setminus M(C_1))$. It is easy to see that if $m_1 \leq_P m_2$, then $C_1 \preceq_P C_2$. Therefore, since $\leq_P$ is a total pre-ordering, two cubes are always comparable.

We now show that $\preceq_P$ is transitive. Let C_1, C_2 and C_3 be three cubes of $\mathcal{C}$ such that $C_1 \prec_P C_2$ and $C_2 \prec_P C_3$. We have to show that $C_1 \prec_P C_3$, i.e. that for every $m_1 \in M(C_1) \setminus M(C_3)$, there exists $m_3 \in M(C_3) \setminus M(C_1)$ such that $m_1 <_P m_3$. Since $\leq_P$ is a total pre-ordering, it is equivalent to show that for every $m_1 \in max_{\leq_P}(M(C_1) \setminus M(C_3))$, there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_1))$ such that $m_1 <_P m_3$. Let $m_1 \in max_{\leq_P}(M(C_1) \setminus M(C_3))$. In the following, we consider two cases: $m_1 \in M(C_1) \setminus M(C_2)$ or $m_1 \in M(C_2)$.

1. If $m_1 \in M(C_1) \setminus M(C_2)$, since $C_1 \prec_P C_2$, there exists $m_2 \in max_{\leq_P}(M(C_2) \setminus M(C_1))$ such that $m_1 <_P m_2$.
   - If $m_2 \in M(C_3)$, then we have $m_2 \in M(C_3) \setminus M(C_1)$ which shows that there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_1))$ such that $m_1 <_P m_3$.
   - If $m_2 \in M(C_2) \setminus M(C_3)$, since $C_2 \prec_P C_3$, there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_2))$ such that $m_1 <_P m_2 <_P m_3$. If $m_3 \in M(C_3) \setminus M(C_1)$, it shows that there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_1))$ such that $m_1 <_P m_3$. Otherwise, if $m_3 \in M(C_1)$, we have $m_3 \in M(C_1) \setminus M(C_2)$. Thus, since $C_1 \prec_P C_2$, there exists $m_2' \in max_{\leq_P}(M(C_2) \setminus M(C_1))$ such that $m_3 <_P m_2'$. Therefore, we have $m_2 <_P m_3 <_P m_2'$ which contradicts the fact that m_2 is maximal in $M(C_2) \setminus M(C_1)$ since $m_2' \in M(C_2) \setminus M(C_1)$.
2. If $m_1 \in M(C_2)$, we have $m_1 \in M(C_2) \setminus M(C_3)$. Therefore, there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_2))$ such that $m_1 <_P m_3$.

   If $m_3 \in M(C_3) \setminus M(C_1)$, it shows that there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_1))$ such that $m_1 <_P m_3$. Otherwise, if $m_3 \in M(C_1)$, we have $m_3 \in M(C_1) \setminus M(C_2)$. Therefore, there exists $m_2' \in max_{\leq_P}(M(C_2) \setminus M(C_1))$ such that $m_1 <_P m_3 <_P m_2'$. If $m_2' \in M(C_3)$, it shows that there exists $m_3 \in max_{\leq_P}(M(C_3) \setminus M(C_1))$ such that $m_1 <_P m_3$. Otherwise, if $m_2' \in M(C_2) \setminus M(C_3)$, since $C_2 \prec_P C_3$, there exists $m_3' \in max_{\leq_P}(M(C_3) \setminus M(C_2))$ such that $m_2' <_P m_3'$. Thus, we have $m_3 <_P m_2' <_P m_3'$ which contradicts the fact that m_3 is maximal in $M(C_3) \setminus M(C_2)$ since $m_3' \in M(C_3) \setminus M(C_2)$.

In conclusion, $\preceq_P$ is transitive, reflexive, and every pair of cubes are comparable, which shows that $\preceq_P$ is a total pre-ordering over $\mathcal{C}$.

Let us show that, for all cubes C_1 and C_2 in $\mathcal{C}$, we have: $C_1 \preceq_P C_2$ if and only if $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2))) \cap C_2 \neq \emptyset$.

Assuming that $C_1 \preceq_P C_2$, let us suppose that we have $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2))) \cap C_2 = \emptyset$. In this case, we have $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2))) \subseteq C_1$. Thus, for every m in $max_{\leq_P}((M(C_1) \setminus$

$M(C_2)) \cup (M(C_1) \setminus M(C_2)))$, it is not possible to have m' in $M(C_2) \setminus M(C_1)$ such that $m \leq_P m'$, which is a contradiction with the fact that $C_1 \preceq_P C_2$.

Conversely, let us assume that $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2))) \cap C_2 \neq \emptyset$, and let m be in $M(C_1) \setminus M(C_2)$. Then, since $\leq_P$ is a complete pre-ordering, we have $m \leq_P m'$ for every m' in $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2)))$. Since the intersection $max_{\leq_P}((M(C_1) \setminus M(C_2)) \cup (M(C_1) \setminus M(C_2))) \cap C_2$ is assumed to be not empty, it follows that $C_1 \preceq_P C_2$. Thus, the proof is complete. ◇

A.2 Proof of Theorem 1

Given a cube $C = \langle D_1, \ldots, D_N, f \rangle$ and a visualization constraint v, let $\langle C^*, S^* \rangle$ with $C^* = \langle D_1^*, \ldots, D_N^*, f^* \rangle$ be the personalized visualization of C computed by function $PersoVisu$. We have to proove that there do not exist a sub-cube $C' = \langle D_1', \ldots, D_N', f' \rangle$ of C and a structure $S' \in \mathcal{S}$ such that $C^* \prec_P C'$ and $v(C', S') = true$.

Assume that such a visualization $\langle C', S' \rangle$ exists. Let $m' \in max_{\leq_P}(\mathcal{M}' \setminus \mathcal{M}^*)$ where $\mathcal{M}' = \bigcup_{i=1}^{N} D_i'$ and $\mathcal{M}^* = \bigcup_{i=1}^{N} D_i^* = \{m_1^*, \ldots, m_L^*\}$ with $m_1^* \geq_P \ldots \geq_P m_L$. Considering the main loop of function $PersoVisu$ (see steps 4-12), m_i^* is the i-th member $m^* \in \mathcal{M}$ that has been selected at step 5 and that has not been rejected at step 10.

Let $\mathcal{M}^+ = \{m_i^* \in \mathcal{M}^* \mid m' <_P m_i^*\}$. Assume that there exists $m_i^* \in \mathcal{M}^+$ such that $m_i^* \notin \mathcal{M}'$. In that case, we have $\mathcal{M}^+ \setminus \mathcal{M}' \neq \emptyset$. It follows that $\mathcal{M}^* \setminus \mathcal{M}' \neq \emptyset$ and that there exists $m^* \in max_{\leq_P}(\mathcal{M}^* \setminus \mathcal{M}')$. Moreover, we have $m^* \geq_P m_i^* >_P m'$, which contradicts the hypothesis that C^* is strictly less interesting that C', i.e. $C^* \prec_P C'$. In conclusion, for every $m_i^* \in \mathcal{M}^+$, we necessarily have $m_i^* \in \mathcal{M}'$, i.e. $\mathcal{M}^+ \subseteq \mathcal{M}'$.

On the other hand, we know that during the computation of C^* and after all members in $\mathcal{M}^+$ have been selected, m' has not been inserted into a set D_i^*. Assume that m' belongs to dimension D_i. Two situations have to be distinguished:

- m' has been selected at step 5, but has been rejected at step 10.
- Another member m'' of the same dimension D_i as m' has been selected before, but has been rejected. In that case, m' has been removed from $\mathcal{M}$, since at step 10, we compute $\mathcal{M} = \mathcal{M} \setminus D_i$.

Denote by m^+ the last member of D_i that has been selected and rejected during the computation of C^* ($m^+ = m'$ or $m^+ = m''$). We now define the sub-cubes C^+ and C'' of C by:

- $C^+ = \langle D_1^+, \ldots, D_N^+, f^+ \rangle$ where for $j = \{1, \ldots, N\}$, $D_j^+ = \{m \in \mathcal{M}^+ \mid m \in D_j\}$ if $j \neq i$, and $D_i^+ = \{m \in \mathcal{M}^+ \mid m \in D_j\} \cup \{m^+\}$.
- $C'' = \langle D_1'', \ldots, D_N'', f'' \rangle$ where for $j = \{1, \ldots, N\}$, $D_j'' = D_j'$ if $j \neq i$, and $D_i'' = (D_i' \setminus \{m'\}) \cup \{m^+\}$.

By construction, we can see that $v(C', S') = v(C'', S') = true$. Moreover, $C^+ \subseteq C''$ since $\mathcal{M}^+ \subseteq \mathcal{M}'$ and $m^+ \in D_i''$. It follows that $(\exists S^+ \in \mathcal{S})(v(C^+, S^+) = true)$ since $v(C'', S'') = true$, $C^+ \subseteq C''$ and v is anti-monotone. That contradicts the fact that m^+ has been rejected during the construction of C^*, and thus, the initial hypothesis that the cube C' exists. In conclusion, it shows that C^* is optimal, which completes the proof. ◇

A.3 Proof of Proposition 2

In order to proof this proposition, we show that the problem defined is more difficult than the Subset Product problem [9].

An instance of the Subset Product problem (SP) is defined by a set $E = \{e_1, \ldots, e_N\}$ of integers e_i ($i = 1, \ldots, N$) and an integer G. Moreover, the question is: is there a subset E' of E such that $\prod_{e_i \in E'} e_i = G$.

On the other hand, an instance of our problem (SVC) can be defined by a set $E = \{e_1, \ldots, e_N\}$ of integers e_i (the cardinalities of the dimensions D_i such that $|D_i| > 1$), an integer K (the number of visible axes), K integers G_k and K subsets X_k of E ($k = 1, \ldots, K$). Moreover, the question is: is there a partition $\{E_1, \ldots, E_K\}$ of E such that for every $k = 1, \ldots, K$, we have $X_k \subseteq E_k$ and $\prod_{e_i \in E_k} \leq G_k$.

In order to prove that SVC is NP-complete, we reduce the SP problem to our problem. Given an instance I_1 of SP, we create an instance I_2 of SVC as follows. Given $N = \prod_{e_i \in E} e_i$, we set $K = 2$, $G_1 = G$, $G_2 = P$ if $P \leq N/G < P + 1$, and $X_1 = X_2 = \emptyset$. Assume that there exists a partition $\{E_1, E_2\}$ of E solution of I_2. Then we have $\prod_{e_i \in E_1} e_i \leq G_1 = G$ and $\prod_{e_i \in E_2} e_i \leq G_2 \leq N/G$. Assume that $\prod_{e_i \in E_1} e_i < G$. It follows that: $N = (\prod_{e_i \in E_1} e_i) \times (\prod_{e_i \in E_2} e_i) < G * N/G = N$, which is not possible and contradicts the hypothesis. Therefore, we necessarily have $\prod_{e_i \in E_1} e_i = G$, which shows that E_1 is a solution of I_1. Conversely, if E' is a solution of I_1, it is easy to see that $\{E', E \setminus E'\}$ is a solution of I_2. Finally, the reduction that we propose to transform an instance of SP into an instance of SVC can be done in polynomial time, which completes the proof. ◇