Securing OLAP Data Cubes Against Privacy Breaches [*]

Lingyu Wang, Sushil Jajodia, and Duminda Wijesekera
Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4444, USA
{lwang3,jajodia,dwijesek}@gmu.edu

Abstract

An OLAP (On-line Analytic Processing) system with insufficient security countermeasures may disclose sensitive information and breach an individual's privacy. Both unauthorized accesses and malicious inferences may lead to such inappropriate disclosures. Existing access control models in relational databases are unsuitable for the multi-dimensional data cubes used by OLAP. Inference control methods in statistical databases are expensive and apply to limited situations only. We first devise a flexible framework for specifying authorization objects in data cubes. The framework can partition a data cube both vertically based on dimension hierarchies and horizontally based on slices of data. We then study how to control inferences in data cubes. The proposed method eliminates both unauthorized accesses and malicious inferences. Its effectiveness does not depend on specific types of aggregation functions, external knowledge, or sensitivity criteria. The technique is efficient and readily implementable. Its on-line performance overhead is comparable to that of the minimal security requirement. Its enforcement requires little modification to existing OLAP systems.

1. Introduction

OLAP (On-line Analytic Processing) is one of the most popular decision support techniques in use today. OLAP enables the exploration of large amounts of data in data warehouses. Aggregations at different levels compose a multi-dimensional *data cube* along *real* enterprise dimensions. By *rolling up* to coarser aggregations, a user may obtain patterns and trends of data. Upon observing an exception to such patterns or trends, the user *drills down* to finer aggregations to catch the outliers. The interactive exploration repeats over *slices* of the data cube until the user ultimately constructs a satisfactory mental image of the underlying data.

Like other technologies, OLAP is also a double-edged sword. Without sufficient security countermeasures, it may become a powerful tool in the hands of malicious users in stealing an enterprise's secrets. Those secrets may include an individual's personal data collected during electronic transactions. Improper disclosures of such data is a threat to the individual's privacy. Unfortunately, most of today's OLAP systems lack effective security countermeasures. Existing mechanisms are usually limited to *data sanitization* and *access control* in relational back-ends. Data sanitization removes explicit identifiers such as names. The data is then deemed *anonymous* and made available to all users. However, data sanitization by itself has long been recognized as insufficient for preserving privacy [15, 25, 26]. Re-identification is possible by combining seemingly unrelated information, such as visiting patterns at different web sites [23]. ROLAP (Relational OLAP) systems rely on their relational back-ends for the control of unauthorized accesses to sensitive data. However, the different data models make it difficult for relational databases to understand the security requirements of OLAP. Moreover, MOLAP (Multi-dimensional OLAP) systems do not have relational back-ends. To overcome such difficulties, we allow authorization objects to be directly specified in data cubes.

OLAP is especially vulnerable to another security threat, malicious *inferences* of sensitive data. OLAP heavily depends on aggregations such as SUMs, MAXs, and so on. Those aggregations hide insignificant details of data and hence accentuate general patterns. However, aggregations do not completely destroy sensitive information. The remaining vestiges, together with *external knowledge* [1], make malicious inferences possible [11]. As a simple example, Bob can infer the amount of Alice's commission from the amount of their total commission, with the external knowl-

∗ This work was partially supported by the National Science Foundation under grant CCR-0113515.

1 The knowledge obtained through channels other than queries [11].

edge of his own commission. Access control cannot capture this inference, as the total salary is a seemingly innocent aggregation. Inference control has been studied in statistical databases and census data from 1970's. However, the complexity results are usually negative in tone [7] for on-line systems, and the proposed methods are rarely seen in commercial products. Many lessons can be learned. For example, most restriction-based methods adopt a *detecting-then-removing* approach. However, the detection of inferences usually demands on-line (that is, after queries are posed) complicated computations over entire data or the bookkeeping of every single answered query. This results in prohibitive on-line performance overhead and storage requirements. Even at such a high cost, the detection is usually only effective in limited situations with unrealistic assumptions. Many methods assume only one specific type of aggregation (for example, SUM-only), and a fixed sensitivity criterion (for example, only exact values causing privacy breach). Compared to statistical databases, OLAP usually demands shorter response time for larger queries. Applying existing techniques to OLAP places us in no better situations than before. Nevertheless, this literature has laid a solid foundation from which our study stems.

The contribution of this work is two-fold. Firstly, we devise a framework for specifying authorization objects in data cubes. The specification is flexible. It partitions the data cube both vertically based on dimension hierarchies and horizontally based on slices of data. The objects in the model are closures of the specified data cube cells. This approach ensures that the finer aggregations implied by the specified ones are also protected. The specification is also distributive over set union, making overlapping objects easy to handle. Secondly, we propose a solution for controlling malicious inferences caused by unprotected coarser aggregations. Instead of detecting inferences, we first prevent complicated inferences through restrictions, and then remove remaining inferences. This novel approach greatly eases inference control. The result is provably secure. It not only eliminates malicious inferences, but also prevents unauthorized accesses by enclosing the result of access control. The method is broadly applicable. It applies to any aggregation functions, external knowledge, and sensitivity criteria, given that some clearly stated properties are satisfied. The technique is efficient and readily implementable. The on-line performance overhead is comparable to that of basic access control, which comprises the minimal security requirement. The off-line complexity and storage requirement are both bounded. The pre-computed result can be enforced with little modification to existing OLAP systems.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 illustrates the data cube model and gives our notations. Section 4 devises a framework for specifying authorization objects in data cubes. Section 5 proposes a solution for controlling malicious inferences in data cubes. Section 6 discusses the implementation options and complexity. Section 7 concludes the paper and describes future directions. All of the proofs can be found in the appendix.

2. Related Work

The need for security and privacy in data warehouses and OLAP has long been identified [4, 29, 30]. However, to the best of our knowledge, solid progress has yet to be made. This is evidenced by most of today's commercial OLAP products, in which the support of access control is limited, and that of inference control is absent [29]. On the other hand, relational databases have mature techniques for both access control and inference control. Access control models regulate direct accesses to sensitive data through owner-specified grants and revokes [18], user-role and role-permission assignments [27], or authorization derivation and resolution rules [20]. Inference control has been extensively studied in statistical databases and census data [1, 12, 35]. The proposed methods can roughly be classified into *restriction-based* techniques and *perturbation-based* techniques. Restriction-based techniques place restrictions on queries, including the minimal number of values aggregated by each query [12], the maximal number of common values aggregated by different queries [13], and the maximal rank of a matrix representing all answered queries [8]. Perturbation-based techniques add random noises to source data [31], query answers [3], or database structures [28]. Our study addresses both unauthorized accesses and malicious inferences of sensitive values, but devotes more effort to the latter, as it actually encloses the former in the specific settings of data cubes.

Related to our work, *cell suppression* is used to protect census data released through statistical tables [9, 10]. Cells containing sensitive COUNTs are suppressed according to a given sensitivity criterion. Possible inferences of the suppressed cells are then detected and removed using linear (or integer) programming-based techniques. While the detection method is effective for two-dimensional cases, it is intractable for three or more dimensional tables even of very small size [10, 11]. We also restrict sensitive cells in data cubes. However, we adopt a fundamentally different approach of preventing complicated inferences instead of detecting them. Our method is thus feasible for data cubes of any size and any dimensional. It is also effective for aggregation functions other than COUNT. *Partitioning* first defines a partition on the set of sensitive data and then restricts queries to aggregate only complete blocks in the partition [7, 36]. As a variation of partitioning, microaggregation replaces clusters of sensitive values with their averages [24, 35]. The result of our method can also be interpreted as partitions of sensitive data. Unlike partitioning or microaggregation, where partitions are especially created for inference control, the focus of our study is to find natural partitions already existing in data cubes. This makes our method readily implementable, since it incurs little modification to existing OLAP systems. The inherent partitions also provide users with a more meaningful answer and hence more semantics of the underlying data.

Parallel to our work, perturbation-based methods have been proposed for *privacy-preserving data mining* [2]. Random noise is added to sensitive values to preserve privacy, while the statistical distribution of unperturbed data can be approximately reconstructed to facilitate data mining tasks. The problem of protecting sensitive data in OLAP is different from that in data mining. The knowledge discovered by data mining, such as classifications and association rules, depend on the distribution models of data. In contrast, OLAP users heavily depend on ad-hoc queries that aggregate small, overlapping sets of values. The precise answers to such queries are not obtainable from distribution models, even if the models can be successfully reconstructed. As suggested by the literature of statistical databases, having both significant noises in sensitive data and unbiased consistent answers to ad-hoc queries is usually infeasible. Our work is not based on perturbation. While all queries are not answered as a result of security requirements, the answer is always precise. *Secure multi-party computation* allows multiple distrusted parties to cooperatively compute aggregations over each other's data [32]. Cryptographic protocols enable each party to obtain the final result with the minimal disclosure of their own data. This problem is different from inference control, because the threat of inferences comes from what users know, not from the way they know it.

As a special case of our problem, inferences of exact values in SUM-only data cubes are studied in [5, 33, 34]. The maximal number of queries that can be answered without causing inferences in a data cube containing no previously known values is given in [5]. A tight upper bound on the number of such known values is given in [34], such that data cubes remain inference-free [34]. *Even range queries* (that is, multi-dimensional axis-parallel boxes containing even number of sensitive values) are not subject to simple inferences, and all inferences can be detected in linear time in the number of queries and values [33]. However, like many other studies on SUM-only inference control in the literature, those results improve our understanding of the problem, but may not be practical. Because in practice an OLAP system is rarely SUM-only, and privacy breach is not only caused by exact values. In this paper, we avoid the pitfall by proposing a method that is independent of the aggregation functions, external knowledge, and sensitivity criteria.

3. The Basic Model of Data Cubes

We closely follow the *data cube* model proposed in [17], which is one of the most popular multi-dimensional models for OLAP. Instead of re-inventing terms, we illustrate the concepts through an example. Then we introduce our notations for those concepts, and state any differences in their interpretation.

Figure 1 depicts a fictitious *data cube*. It has two *dimensions*: $time$ and $organization$. The $time$ dimension has three *attributes*: $quarter$, $year$, and all [2]. The $organization$ dimension has four attributes: $employee$, $department$, $branch$, and all. The attributes of each dimension are partially ordered (totally ordered in this special case) by the *dependency relation* $\preceq$ into a *dependency lattice* [19]. That is, $quarter \preceq year \preceq all$ for the $time$ dimension and $employee \preceq department \preceq branch \preceq all$ for the $organization$ dimension. The product of the two lattices gives the dependency lattice of cuboids. Each element of this dependency lattice is a tuple $< T, O >$, where T is an attribute of the $time$ dimension and O is an attribute of the $organization$ dimension. Attached to each such tuple $< T, O >$ is an empty two-dimensional array, namely, a *cuboid*. Each *cell* of the cuboid $< T, O >$ is also a tuple $< t, o >$, where t and o are *attribute values* of the attribute T and O, respectively. The dependency relation between cuboids extend to their cells. For example, the cuboid $< year, employee >$ depends on the cuboid $< quarter, employee >$, hence a cell $< Y1, Bob >$ of the former also depends on the cells $< Q1, Bob >$, $< Q2, Bob >$, $< Q3, Bob >$, and $< Q4, Bob >$ of the latter. Similarly, the cell $< Q1, Book >$ depends on the cells $< Q1, Bob >$, $< Q1, Alice >$, $< Q1, Jim >$, and $< Q1, Mallory >$ (suppose the book department only has those four employees). Hence, all cells also form a dependency lattice.

A *fact table* is a relational table having the schema $(quarter, employee, commission)$. The fact table is used to populate the data cube with values of the *measure* attribute $commission$. Each record in the fact table, a triple (q, e, m), is used to populate a cell $< q, e >$ of the *core cuboid* $< quarter, employee >$, where q, e, and m are values of the attributes $quarter$, $employee$, and $commission$, respectively. Some cells of $< quarter, employee >$ remain empty (or having the $NULL$ value), if corresponding records are absent in the fact table. All cuboids are then populated using the *aggregation function* SUM (for simplicity purposes, only three populated cuboids are shown in Figure 1). For example, in the cuboid $< year, employee >$, a cell $< Y1, Bob >$ takes the value 8500, which is the to-

2 We regard all as a special attribute having one attribute value ALL, which depends on all other attribute values.

tal amount of the four cells it depends on, $< Q1, Bob >$, $< Q2, Bob >$,$< Q3, Bob >$, and $< Q4, Bob >$. An empty cell is deemed as zero in the aggregation. As another example, the cuboid $< all, employee >$ (its cells are not shown in Figure 1) can be computed from either the core cuboid $< quarter, employee >$ or the cuboid $< year, employee >$, because it depends on both.

Assume a fixed order among dimensions, among attributes of each dimension, and among values of each attribute. Denote the i^{th} ($1 \leq i \leq k$ for some fixed k) dimension D_i as a set of attributes $D_i = \{d_i^j : 1 \leq j \leq \mid D_i \mid\}$ (superscripts and subscripts will be omitted whenever appropriate), where $\mid D_i \mid$ denotes the cardinality of D_i. The data cube is the collection of all cuboids, denoted by the Cartesian product of the k dimensions, $\mathcal{L} = \prod_{i=1}^{k} D_i$. Each cuboid $\vec{c} \in \mathcal{L}$ is a k-tuple of attributes $\vec{c} =< d_1, d_2, \ldots, d_k >$, with $d_i \in D_i$. Similarly, each attribute d_i is also viewed as a set of attribute values. A cuboid $\vec{c} =< d_1, d_2, \ldots, d_k >$ is thus a collection of cells [3], denoted by the Cartesian product$\prod_{i=1}^{k} d_i$. Each cell $\vec{t} \in \vec{c}$ (here $\vec{c}$ means $\prod_{i=1}^{k} d_i$) is a k-tuple of attribute values. Use $\mathcal{A} = \bigcup_{\vec{c} \in \mathcal{L}} \vec{c}$ for the set of all cells in a data cube $\mathcal{L}$. Use $< \mathcal{L}, \preceq >$ for the dependency lattice of cuboids. Then $< \mathcal{A}, \preceq >$ is the dependency lattice of all cells. The content of a cell, an aggregation of the measure attribute, is not explicitly denoted. Instead we use a cell to interchangeably refer to both the cell itself and its content, when the actual meaning is clear from context.

We borrow some terms from lattices [14]. In a lattice $< L, \preceq >$ (such as $< \mathcal{L}, \preceq >$ and $< \mathcal{A}, \preceq >$), any $x \in L$ and $y \in L$ is *non-comparable*, if neither $x \preceq y$ nor $y \preceq x$ hold; they are comparable, otherwise. Any $y \in L$ is an *ancestor* of $x \in L$ (and dually x is a *descendant* of y) if $y \preceq x$ holds [4]. For any $L' \subseteq L$, the *GLB* (greatest lower bound) of L' is any $x \in L$ satisfying that the condition $x \preceq y$ for any $y \in L'$ holds, and no descendant of x satisfies the condition. The dual concept of GLB is *LUB* (lowest upper bound). The GLB and LUB of any (set of) two elements $x \in L$ and $y \in L$ are called their *meet* and *joint*, and denoted as $x \wedge y$ and $x \vee y$, respectively. x is a *maximal* (or *minimal*) element of any $L' \subseteq L$, if $x \preceq y$ (or $y \preceq x$) and $y \in L$ implies $y = x$.

Different from [17], we regard a cell as empty if and only if its value is known from external knowledge. An empty cell has the $NULL$ value in [17]. In our study, a cell having the $NULL$ value may be non-empty, if users do not know this fact; conversely, any previously known cell is empty regardless of its value. Because from the security point of view, the specific value of a cell not longer matters, if it has been learned by the malicious user before he/she poses queries. Another difference is that we consider an attribute as a measure attribute(such as the commission in Figure 1) only if it is sensitive. In [17], any numerical attribute may be a measure attribute.

4. Specifying Authorization Objects in Data Cubes

In this section, we devise a framework for specifying authorization objects in data cubes. An authorization is usually a triple: $(object, subject, (signed)action)$, indicating that the subject is allowed (or prohibited, depending on the sign of action) to execute the action on the object [20]. We only consider one type of action, *read*. Because we regard the confidentiality of data as the major security concern in OLAP. Other actions such as updates are relatively infrequent, and are accessible to only a few privileged operators. Subjects are users or user groups. We assume subjects do not collude, but users may. That is, a group of users inclined to collusion should be regarded as a single subject. We assume an open policy, where only prohibitions are specified, and permissions are implied by the absence of prohibitions [5].

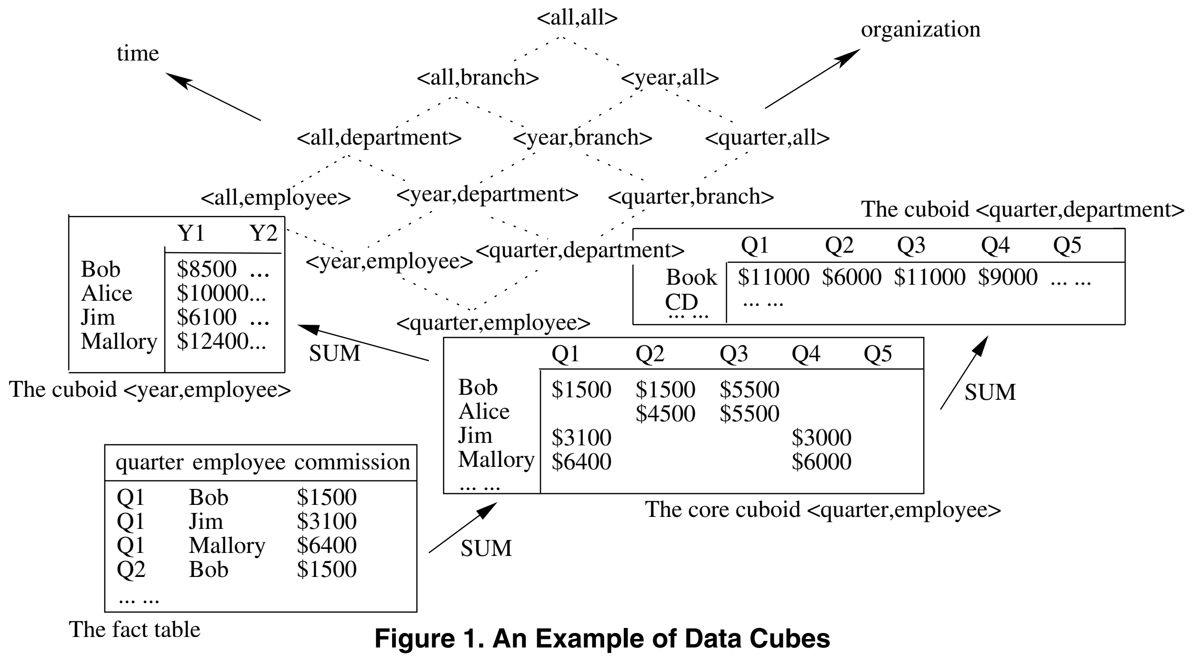
The major difference between the authorization requirements of OLAP and that of relational databases lies in their authorization objects. This is decided by their different data models. In a relational model, typical objects include tables, records in a table, or fields of a record. By partitioning tables vertically into records and horizontally into fields, authorizations can be more precise. An analogous spectrum of objects needs to be defined in a multi-dimensional model, such as a data cube. Two independent aspects of a data cube come across. Firstly, a data cube includes aggregations at different granularity levels, as in Figure 1. A subject's responsibilities may only entitle him/her to the aggregations above a certain level. Anything below is either sensitive or irrelevant, and hence should be protected according to the principle of least privilege (that is, only what is needed is permitted). Secondly, a data cube can be divided into slices by selections over one or more dimensions. Similarly, the least privilege principle also requires a subject to be confined to some slices based on his/her needs. Example 4.1 illustrates such requirements.

Example 4.1 *Suppose an analyst Eve is invited to analyze the data cube in Figure 1. However, two authorization requirements exist. Firstly, due to privacy concerns, Eve should not learn the value of any employee's commission, although she may access the values of departments*

3 Here the k-tuple of attributes $< d_1, d_2, \ldots, d_k >$ can be viewed as a label of the collection of cells it represents.

4 Notice that in our graphical representation, such as in Figure 1, the descendants are always above the ancestors.

5 Our framework can be easily extended to specify both prohibitions and permissions, using techniques such as FAF [20].



	Y1	Y2
Bob	$8500	...
Alice	$10000...	
Jim	$6100	...
Mallory	$12400...	

	Q1	Q2	Q3	Q4	Q5
Book	$11000	$6000	$11000	$9000	... ...
CD	... ...				
... ...					

	Q1	Q2	Q3	Q4	Q5
Bob	$1500	$1500	$5500		
Alice		$4500	$5500		
Jim	$3100			$3000	
Mallory	$6400			$6000	
... ...					

quarter	employee	commission
Q1	Bob	$1500
Q1	Jim	$3100
Q1	Mallory	$6400
Q2	Bob	$1500
... ...		

Figure 1. An Example of Data Cubes

or branches. Secondly, any data no older than the quarter $Q5$ *should not be used for analysis. Clearly, Eve should not access the three cuboids* $< quarter, employee >$, $< year, employee >$, *and* $< all, employee >$. *Moreover, she is also prohibited from reading cells such as* $< Y2, Book >$ *(now shown in Figure 1), since the data is no older than* $Q5$ *(suppose* $Y2$ *depends on* $Q5$, $Q6$, $Q7$, *and* $Q8$*). Notice the difference between the two requirements. The former divides the* $organization$ *dimension into two parts:* $employee$ *and* $\{department, branch, all\}$, *based on the dimension hierarchy. The latter partitions the* $time$ *dimension into* older than Q5 *and* no older than Q5, *based the attribute values.*

In Example 4.1, suppose an administrator specifies the first requirement with a cuboid $< all, employee >$. The other two cuboids $< year, employee >$ and $< quarter, employee >$ are implied by this specification. Because the specified cuboid $< all, employee >$ can be computed from either of them, if not protected. However, the administrator should not be burdened to ensure that all implied cuboids be included by a specification. Instead, we allow an *l-specification* (that is, level specification) S_l to be an arbitrary set of cuboids. The *closure* $Below(S_l)$ of an l-specification S_l is formalized in Definition 1. A closure gives all and only the cuboids to be protected. The closure can be regarded as a *vertical partition* of the data cube.

Definition 1 *In any data cube* $\mathcal{L}$, *a function* $Below(.) : 2^{\mathcal{L}} \rightarrow 2^{\mathcal{L}}$ *is defined as* $Below(S_l) = \{\vec{c}:$ *there exists* $\vec{c}_s \in S_l,$ *such that* $\vec{c} \preceq \vec{c}_s$ *holds*$\}$. *We say a set of cuboids* S_l *is an* **l-specification** *(that is, level specification), and* $Below(S_l)$ *is the* **closure** *of* S_l.

Example 4.2 *The requirement that Eve should not learn an employee's information can now be represented by the closure* $Below(< all, employee >)$ *of the l-specification* $< all, employee >$.

The second requirement in Example 4.1 requires a *horizontal partition* of the data cube. Generally, an *s-specification* (that is, slice specification) r is an arbitrary set of cells (usually, but not necessarily, in one cuboid). The *slice* $Slice(r)$ of an s-specification r, as formalized in Definition 2, includes all and only the cells comparable to at least one cell in the s-specification.

Definition 2 *In any data cube* $\mathcal{L}$ *with the set of all cells* $\mathcal{A}$ *(that is,* $\mathcal{A} = \bigcup_{\vec{c} \in \mathcal{L}} \vec{c}$*), define a function* $Slice(.) : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ *as* $Slice(r) = \{\vec{t}:$ *there exists* $\vec{t}_1 \in r$ *such that* $\vec{t} \preceq \vec{t}_1$ *or* $\vec{t}_1 \preceq \vec{t}$ *holds* $\}$. *We say any* $r \subseteq \mathcal{A}$ *is an* **s-specification** *(that is, slice specification), and* $Slice(r)$ *is the* **slice** *of* r.

Example 4.3 *The requirement that Eve should not learn anything no older than* $Q5$ *can be represented by the slice* $Slice(\{< q, e >: q \in quarter, e \in employee, q \geq Q5\})$.

The authorization object may be specified by different administrators over time. Hence, for each subject there may be more than one pair of s-specification and l-specification. Without loss of generality, suppose two pairs (r_1, S_1) and (r_2, S_2) are specified for a subject, where r_1 and r_2 are s-specifications, and S_1 and S_2 are l-specifications. Further suppose that neither $Below(S_1) \subseteq Below(S_2)$ nor

$Below(S_2) \subseteq Below(S_1)$ holds. Then the intersection of the two slices $Slice(r_1) \cap Slice(r_2)$ corresponds to a new closure $Below(S_1 \cup S_2)$. It would be prohibitive if such a new closure needs to be computed for every intersection of slices. Because there may be an exponential (in the number of slices) number of intersections. However, the function $Below()$ is distributive over set union, as stated in Proposition 1. This desired property allows the intersections of slices to be ignored in controlling accesses to an object, given that the open policy is properly enforced (that is, a request is permitted only if no prohibition exists). Similarly, the *boundary* cells among slices can also be ignored. For example, a cell $< ALL, Book >$ (not shown in Figure 1) belongs to both the slice *older than Q5* and the slice *no older than Q5* (since the value ALL depends on all other values). The request for such a cell will be denied, because a prohibition exists in the second slice, according to Example 4.3.

Proposition 1 *In any data cube $\mathcal{L}$, $Below(S_1 \cup S_2) = Below(S_1) \cup Below(S_2)$ holds for any $S_1 \in \mathcal{L}$ and $S_2 \in \mathcal{L}$.*

We define authorization objects in Definition 3 by combining closures and slices. An example of objects is then given in Example 4.4. Given an authorization $(Object(O), subject)$, the *basic access control* denies accesses to any protected cell. Specifically, a request for a cell $\vec{t}$ is denied, if $\vec{t} \in Object(O)$ holds; the request is granted, otherwise. However, while the basic access control comprises the *minimal security requirement*, it is ineffective to protect the object because of malicious inferences, as illustrated in Section 5.1. Moreover, we shall show in Section 5.2.2 that the object of the basic access control is actually enclosed by that of the inference control. Hence, we conclude this section without further addressing the basic access control.

Definition 3 *In any data cube $\mathcal{L}$ with the set of all cells $\mathcal{A}$, define a function $Object(.) : 2^{2^{\mathcal{A}} \times 2^{\mathcal{L}}} \rightarrow 2^{\mathcal{A}}$ as $Object(\{(r_i, S_i) : 1 \leq i \leq n\}) = \{\vec{t} : \vec{t} \in Slice(r_i)$ and $\vec{t} \in \vec{c}$ both hold, for some $1 \leq i \leq n$ and $\vec{c} \in Below(S_i)\}$. We say $Object(O))$ is an* **authorization object** *(or simply an object) specified by the pairs of s-specification and l-specification $O = \{(r_i, S_i) : 1 \leq i \leq n\}$. We say any cell $\vec{t} \in Object(O)$ is* **protected**, *and any cell $\vec{t} \in \mathcal{A} \setminus Object(O)$* **unprotected**.

Example 4.4 *Following previous examples, the object for Eve is $Object(O)$, where $O = \{(r_1, S_1), (r_2, S_2)\}$. The specifications are $r_1 =< ALL, ALL >$, $S_1 =< all, employee >$, $r_2 = \{< q, e >: q \in quarter, e \in employee, q \geq Q5\}$, and $S_2 =< all, all >$ (notice that $< ALL, ALL >$ is a cell and $< all, all >$ is a cuboid). Examples of protected cells include $< Y1, Bob >$ (in both $Slice(r_1)$ and $Below(S_1)$), $< ALL, Book >$(in both $Slice(r_2)$ and $Below(S_2)$) and $< Y2, Alice >$ (in both $Slice(r_1) \cap Slice(r_2)$ and $Below(S_1 \cup S_2)$).*

5. Controlling Malicious Inferences in Data Cubes

The definition of an authorization object prevents any protected cell from being computed from *below* (that is, from its ancestors). However, in many cases a protected cell can be inferred from *above* (that is, from its descendants). Such inferences can easily infiltrate the line of defense established by the basic access control. In this section, we address inference control in data cubes. First, in Section 5.1 we illustrate different kinds of inferences in data cubes through examples. Then, in Section 5.2 and 5.3 we propose methods to prevent and eliminate inferences.

5.1. Inferences in Data Cubes

In data cubes, an *inference* occurs when sensitive information about the values of some protected cells are derived from the values of their unprotected descendants as well as from external knowledge. The inference thus depends on several factors. That is, the *source* and *target*, which is the set of unprotected cells causing the inference and the set of protected cells being inferred, respectively; the *aggregation function*, which is used to compute the value of each cell; the *sensitive criterion*, which decides what information is sensitive; and *external knowledge*, which has been learned through channels other than queries. Most inference control methods adopt a *detecting-then-removing* approach. Such an approach usually assumes a specific type of aggregation functions, sensitivity criteria, and external knowledge, in order to reduce the complexity of detecting inferences. For example, only SUMs are allowed; only the exact values are sensitive; or subjects know nothing about the data type of sensitive attributes. Extending a detection method to remove such assumptions is usually infeasible. For example, Chin shows that detecting inferences by *auditing* [8] is polynomial for both SUM-only and MAX-only cases. However, it is NP-hard for the case of both SUM and MAX [6]. Even for the SUM-only case, the detection becomes infeasible, if both exact sensitive values and the small intervals enclosing such values are considered as sensitive [10, 22], or when a subject knows that the sensitive attributes are binary [21]. Unfortunately, a practical OLAP system does not restrict its users to SUM-only, nor hides from users the data type of an attribute. Different systems may also adopt different sensitivity criteria. Hence, the detecting-then-removing approach may not be practical for data cubes.

However, a special kind of inferences, *1-d inferences* (that is, one-dimensional inferences), can be effectively detected in data cubes. As illustrated in Example 5.1, in 1-d inferences, the source, as well as the target, is the subset of a single cuboid. In such a case, the cells in the source depend on disjoint sets of cells in the target. In Example 5.1, each cell in the source $< quarter, department >$ depends on a different column (in the viewable area) of the target $< quarter, employee >$. Informally, this property of 1-d inferences implies that the detection of 1-d inferences can be *partitioned*, in the sense that the cells in the source do not help each other in gaining inferences (a more formal statement is given in Definition 4). Hence, 1-d inferences can be detected by evaluating each cell in the source against its ancestors in the target, using any given sensitivity criteria. This simple detection procedure is effective in most cases.

Example 5.1 (1-d Inferences) *Suppose that in Figure 1 the cuboid $< quarter, employee >$ is protected from Eve, but its descendant $< quarter, department >$ is unprotected. Further suppose Eve already knows about the empty cells, and the fact that Bob and Alice are taking the same amount of commission in Q3. Eve can then infer the cell $< Q3, Bob >$ and $< Q3, Alice >$ as 5500, which is half the amount 11000 of $< Q3, Book >$.*

The detection becomes expensive and less effective for the complementary case, *m-d* inferences (that is, multi-dimensional inferences). In m-d inferences, the source causes inferences to the target, but the intersection of the source with any single cuboid does not. m-d inferences come from multiple non-comparable descendants of the target. Example 5.2 illustrates two-dimensional inferences in SUM-only data cubes [6]. Example 5.3 and 5.4 illustrate m-d inferences in MAX-only data cubes and in the data cubes where SUM, MAX, and MIN are all allowed. From those examples, two observations are possible. Firstly, different aggregation functions require different detection methods. Secondly, unlike the detection of 1-d inferences, the detection of m-d inferences is not partitionable.

Example 5.2 (m-d Inferences in SUM-only Data Cubes) *Suppose that the external knowledge about identical values cannot be learned, and hence the inference described in Example 5.1 can no longer by achieved by Eve. Assume Eve has accesses to the cuboid $< quarter, department >$ as well as to $< year, employee >$. Notice that the two cuboids are both free of 1-d inferences, because each of their cells depends on two or more cells in the target $< quarter, employee >$ (suppose only exact values are sensitive). However, m-d inferences are possible in the following way. Eve first sums the two cells $< Y1, Bob >$ and $< Y1, Alice >$ in the cuboid $< year, employee >$, then subtracts from the result 18500 the two cells $< Q2, Book >$ (that is, 6000) and $< Q3, Book >$ (that is, 11000) in the cuboid $< quarter, department >$. The protected cell $< Q1, Bob >$ is then inferred as 1500.*

6 Three or more dimensional inferences can be easily constructed, although we do not give examples here due to space limitations.

Example 5.3 (m-d Inferences in MAX-only Data Cubes) *Suppose the external knowledge about empty cells is now prevented. Now the cuboid $< quarter, employee >$ seems to Eve as being full of unknown values. Such a* full SUM-only data cube *is safe from m-d inferences [34]. However, the following m-d inference is possible with MAXs (the MAXs are not shown in Figure 1). Eve applies the MAX to $< Y1, Mallory >$ and $< Q4, Book >$ and gets 6400 and 6000 as the result, respectively. She can then infer that one of the three cells $< Q1, Mallory >$, $< Q2, Mallory >$, and $< Q3, Mallory >$ must be 6400, because $< Q4, Mallory >$ must be no greater than 6000. Similarly, Eve concludes that $< Q2, Mallory >$ and $< Q3, Mallory >$ cannot be 6400, either. The protected cell $< Q1, Mallory >$ is then successfully inferred as 6400.*

Example 5.4 (m-d Inferences with SUM, MAX and MIN) *Finally, suppose Eve can ask SUMs, MAXs and MINs. By Example 5.3, $< Q1, Mallory >$ is 6400. Eve then applies to $< Y1, Mallory >$ MAX,MIN, and SUM, and gets 6400,6000, and 12400 as the answers (the MAXs and MINs are not shown in Figure 1). Eve infers $< Q2, Mallory >$,$< Q3, Mallory >$, and $< Q4, Mallory >$ must be 6000 and two zeroes, although she does not know exactly which is 6000. Eve then applies MAX,MIN, and SUM to $< Q2, Book >$, $< Q3, Book >$ and $< Q4, Book >$, whose result tells Eve the following facts. In $< quarter, employee >$, two cells in Q2 are 1500 and 4500; in Q3 are 5500 and 5500; in Q4 are 3000 and 6000; and the rest are all zeroes, although she cannot match the values to exact cells yet. Eve then conclude that $< Q4, Mallory >$ must be 6000, because the values in Q3 and Q2 cannot be. Similarly, Eve can infer $< Q4, Jim >$ as 3000, and consequently all cells in the cuboid $< quarter, employee >$, and hence the whole data cube, even without any external knowledge.*

In a few special cases, such as in the SUM-only or MAX-only data cubes in Example 5.2 and 5.3, detecting m-d inferences is possible [8, 33]. To the best of our knowledge, no known methods can effectively detect m-d inferences in the more general case. Moreover, even for those special cases, the computational complexity and storage requirement render the detection infeasible. Because unlike 1-d inferences, any cells in a source may help each other in gaining m-d inferences. To make it worse, any cells answered in the past may also help the currently requested ones. Consequently, a detecting method must either keep track of all answered cells for each subject or examine the whole data cube for even a few requested cells.

5.2. Preventing Multi-Dimensional Inferences

From the examples and discussions in Section 5.1, m-d inferences are clearly the major obstacle in making inference control practical for data cubes. In this section, we propose a method to prevent m-d inferences, instead of detecting and then removing them. This novel approach makes our method effective for different aggregation functions, sensitivity criteria, and external knowledge. It also reduces the complexity of the method to a practical level.

5.2.1. Assumptions Although we do not assume specific aggregation functions, sensitivity criteria or external knowledge, we require some assumptions. They are stated as the three conditions in Definition 4. Although those conditions are only required for sensitivity criteria, they also reflect the assumptions about aggregation functions and external knowledge. The first condition says that if a source S includes both a cell and all its ancestors in a cuboid, then removing the cell from S does not change its sensitivity with respect to the target T. This is reasonable, because the cell can be computed from the ancestors by assuming *distributive* aggregation functions [17] such as SUM, MAX, and MIN. For a non-distributive aggregation function, we can keep the assumption by replacing the function with an *intermediate functions*. For example, AVG is not distributive, but the pair $(SUM, COUNT)$ is distributive, and AVG can certainly be computed from $(SUM, COUNT)$. As another example, the identity function, which is clearly distributive, can act as an intermediate function for any aggregation function. The second condition in Definition 4 says that inferences come from descendants only. This does not hold if external knowledge can relate two cells in non-comparable cuboids. In that case, we could regard one as empty (that is, known from external knowledge) if the other is unprotected. The third condition basically generalizes the observation we made about Example 5.1. That is, inferences can be partitioned, if the source S can be divided into blocks that depend on disjoint sets of cells in the target T.

Definition 4 *In any data cube $\mathcal{L}$ with the set of all cells $\mathcal{A}$, any binary function $Sensitive(.) : 2^{\mathcal{A}} \times 2^{\mathcal{A}} \rightarrow \{TRUE, FALSE\}$ is a* **sensitivity criterion**, *if the following all hold:*

1. *$Sensitive(S,T) = Sensitive(S \setminus \{\vec{t_1}\}, T)$, for any cell $\vec{t_1}$ in a cuboid $\vec{c_1}$, if there exists $\vec{c} \prec \vec{c_1}$ satisfying $\{\vec{t} : \vec{t} \in \vec{c}, \vec{t} \preceq \vec{t_1}\} \subseteq S$.*
2. *$Sensitive(S,T) = Sensitive(S \setminus \vec{c_1}, T)$, for any cuboid $\vec{c_1}$, if $\vec{c_1}$ is non-comparable to any cuboid $\vec{c}$ satisfying $\vec{c} \cap T \neq \phi$.*
3. *$Sensitive(S,T) = Sensitive(S_1, T \cap T_1)) \wedge Sensitive(S_2, T \cap T_2))$, if $S_1 \cup S_2 = S$ and $T_1 \cap T_2 = \phi$, where $T_1 = \{\vec{t} : \vec{t} \preceq \vec{t_1}$ holds for some $\vec{t_1} \in S_1\}$ and $T_2 = \{\vec{t} : \vec{t} \preceq \vec{t_2}$ holds for some $\vec{t_2} \in S_2\}$*

For any set of cells S and T, if $Sensitive(S,T) = TRUE$, we say S is **sensitive** *with respect to T; S is* **insensitive**, *otherwise.*

5.2.2. A Special Case We first consider a special case, where the s-specification r is always a complete cuboid instead of a subset of it. The object $Object((r, S_l))$ is thus simply (the union of) protected cuboids in $Below(S_l)$ (we shall omit *the union* and simply say a set of cuboids whenever possible). Example 5.5 describes such an object. Consider the set of protected cuboids $Below(S_l)$ as the target, and the set of unprotected cuboids $\mathcal{L} \setminus Below(S_l)$ as the source. Our objective is to find a subset of the source that is *free of m-d inferences* to the target and is at the same time *maximal*.

Example 5.5 *Figure 2 shows part of the dependency lattice of a four-dimensional data cube with a Hasse diagram [14]. Each a^i, b^i, c^i, and d^i is an attribute. The dot lines denote the dependency relation (those implied by the reflexivity and transitivity of the dependency relation are omitted). Let $S_l = \{< a^1, b^1, c^2, d^2 >, < a^1, b^2, c^1, d^2 >\}$), the lower curve in the solid line depicts an object $Below(S_l)$. All cuboids below the lower curve are protected, and those above are unprotected. Apparently, m-d inferences are possible from unprotected cuboids to protected ones, such as from $\{< a^1, b^2, c^2, d^2 >, < a^2, b^1, c^2, d^2 >\}$ to $< a^1, b^1, c^2, d^2 >$. Because the former includes two non-comparable descendants of the latter.*

Firstly, to simplify the tasks, not all cuboids in the source need to be considered for m-d inferences to each cuboid in the target. According to the first and second conditions in Definition 4, for each cuboid $\vec{c_t}$ in the target $Below(S_l)$, we can reduce the source to a minimal subset without changing the sensitivity. Specifically, we remove a cuboid if it either is non-comparable to $\vec{c_t}$, or is a descendant of other cuboids in the source. The result includes only the minimal elements in the set of descendants of $\vec{c_t}$, namely, the *basis* $Basis(S)$ of the source S with respect to the cuboid $\vec{c_t}$, as formalized in Definition 5. An example of the basis is given in Example 5.6.

Definition 5 *In any data cube $\mathcal{L}$, we define a function $Basis(.) : 2^{\mathcal{L}} \times \mathcal{L} \rightarrow 2^{\mathcal{L}}$ as $Basis(S, \vec{c_t}) = \{\vec{c} : \vec{c} \in S, \vec{c_t} \preceq \vec{c}, (\vec{c_1} \in S \wedge \vec{c_1} \preceq \vec{c})$ implies $\vec{c_1} = \vec{c}\}$. We say $Basis(S, \vec{c_t})$ is the* **basis** *of the source S with respect to a cuboid $\vec{c_t}$ in the target.*

Example 5.6 *From Example 5.5, we have $Basis(\mathcal{L} \setminus Below(S_l), < a^1, b^1, c^2, d^2 >) = \{< a^1, b^2, c^2, d^2 >$*

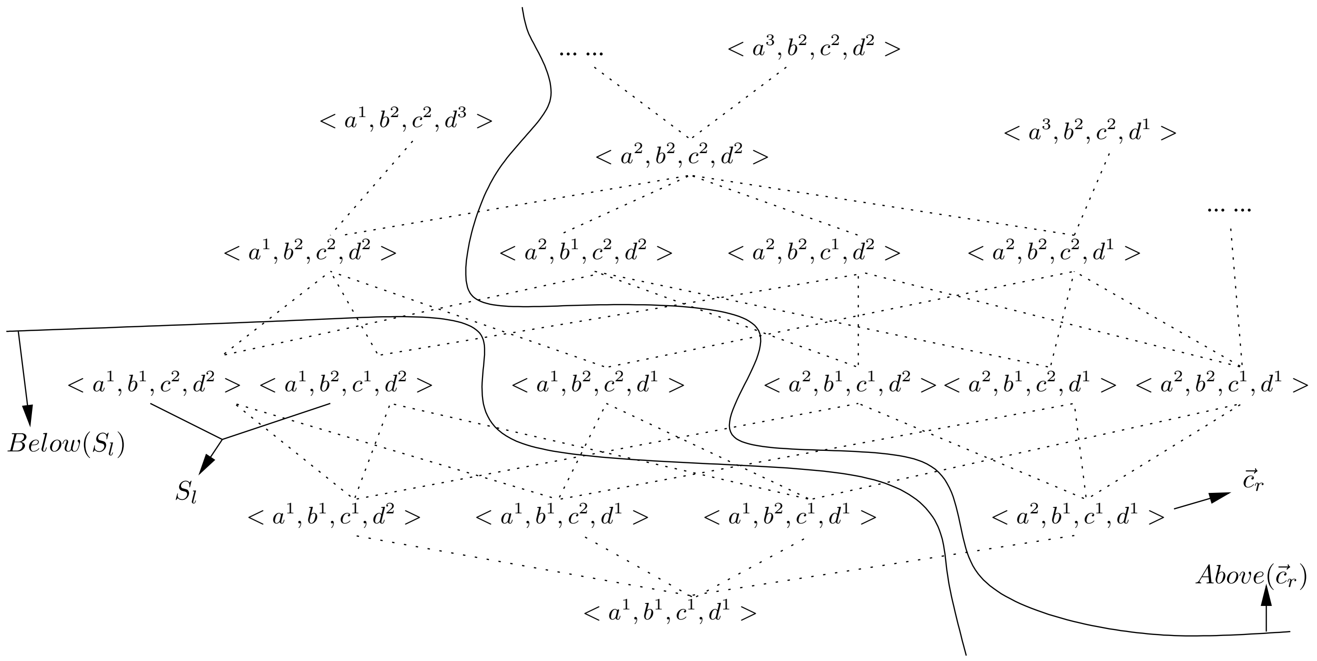


Figure 2. An Example of Preventing m-d Inferences

$, < a^2, b^1, c^2, d^2 >\}$. That is, for inferences to the protected cuboid $< a^1, b^1, c^2, d^2 >$, we only need to consider its two non-comparable descendants $< a^1, b^2, c^2, d^2 >$ and $< a^2, b^1, c^2, d^2 >$.

Because a basis includes only non-comparable cuboids, m-d inferences are possible from the source S to $\vec{c}_t$ only if $Basis(S, \vec{c}_t)$ includes more than one cuboid. For example, m-d inferences are possible in Example 5.6, since the basis includes two cuboids. Conversely, we can construct a subset free of m-d inferences to $\vec{c}_t$ by *growing* from a single *root*. More specifically, we first choose an unprotected cuboid $\vec{c}_r$ satisfying $\vec{c}_r \succeq \vec{c}_t$, then we include all the descendants of $\vec{c}_r$. The result must satisfy that its basis with respect to $\vec{c}_t$ includes only one cuboid, the root $\vec{c}_r$. Hence, the result is free of m-d inferences to $\vec{c}_t$. This process is formalized as a function $Above()$ in Definition 6 [7]. An example is then given in Example 5.7.

Definition 6 *In any data cube $\mathcal{L}$, we define a function $Above(.) : \mathcal{L} \rightarrow 2^{\mathcal{L}}$ as $Above(\vec{c}_r) = \{\vec{c} : \vec{c}_r \preceq \vec{c}\}$. We say the cuboid $\vec{c}_r$ is the* **root** *of $Above(\vec{c}_r)$.*

Example 5.7 *In Figure 2, the cuboids above the upper curve in the solid line comprise $Above(\vec{c}_r)$, where the root $\vec{c}_r =< a^2, b^1, c^1, d^1 >$ is an unprotected descendant of the core cuboid $< a^1, b^1, c^1, d^1 >$. Clearly, no m-d inference is possible from $Above(< a^2, b^1, c^1, d^1 >)$ to $< a^1, b^1, c^1, d^1 >$, since the basis $Basis(Above(< a^2, b^1, c^1, d^1 >), < a^1, b^1, c^1, d^1 >)$ includes only the root itself.*

However, we need to prevent m-d inferences to not only the cuboid $\vec{c}_t$, but also all cuboids in the target. Consider another protected cuboid $\vec{c}_1$ that is non-comparable to the root $\vec{c}_r$. Then $Basis(Above(\vec{c}_r), \vec{c}_1)$ must be different from $Basis(Above(\vec{c}_r), \vec{c}_t)$, because at least the root $\vec{c}_r$ is not included by the former. The question arises: is $Basis(Above(\vec{c}_r), \vec{c}_1)$ also a singleton set? Lemma 1 shows this is indeed the case. Moreover, the cuboid included by $Basis(Above(\vec{c}_r), \vec{c}_1)$ is the joint $\vec{c}_1 \vee \vec{c}_r$. Hence, we have successfully obtained a subset $Above(\vec{c}_r)$ of the source $\mathcal{L} \setminus Below(S_l)$ that is free of m-d inferences to the target $Below(S_l)$. Moreover, Lemma 1 extends the target from $Below(S_l)$ to $\mathcal{L} \setminus Above(\vec{c}_r)$. This is important because otherwise m-d inferences may first get to the unprotected cuboids between $Below(S_l)$ and $Above(\vec{c}_r)$, and then continue to infer the protected cuboids in $Below(S_l)$.

Lemma 1 *In any data cube $\mathcal{L}$, given any l-specification S_l and a root $\vec{c}_r \in \mathcal{L} \setminus Below(S_l)$, for any cuboid $\vec{c}_t \in \mathcal{L} \setminus Above(\vec{c}_r)$, $Basis(Above(\vec{c}_r), \{\vec{c}_t\}) = \{\vec{c}_r \vee \vec{c}_t\}$ holds.*

Example 5.8 *In Figure 2, now consider the cuboids above the upper curve as the source, and those below the upper curve as the target. From any cuboid in the target, exactly one dot line goes into the source. Each such line points to the joint of this cuboid and the root $< a^2, b^1, c^1, d^1 >$. For example, a line goes*

7 The concept is known as an *ideal* in lattice theory [14].

from the leftmost cuboid $< a^1, b^1, c^2, d^2 >$ to its joint with the root, $< a^2, b^1, c^2, d^2 >$ (another line points to $< a^1, b^2, c^2, d^2 >$, which is not in the source). Similarly, from $< a^1, b^2, c^2, d^2 >$, a line points to $< a^2, b^2, c^2, d^2 >$ but other lines do not point to the source. Those observations indicate the absence of m-d inferences.

Next we consider the maximality of this result. The maximality is achieved by choosing a minimal root. More precisely, we let $\vec{c}_r$ be a minimal element of the set of unprotected cuboids $\mathcal{L} \setminus Below(S_l)$. Lemma 2 then states that for any unprotected cuboid not included by $Above(\vec{c}_r)$, its meet with the root must be a protected cuboid. Hence, this cuboid together with the root may cause m-d inferences to their meet. That is, $Above(\vec{c}_r)$ is indeed maximal.

Lemma 2 *In any data cube $\mathcal{L}$, given any l-specification S_l and a minimal element of $\vec{c}_r \in \mathcal{L} \setminus Below(S_l)$, $\vec{c}_1 \wedge \vec{c}_r \in Below(S_l)$ holds for any $\vec{c}_1 \in \mathcal{L} \setminus (Below(S_l) \cup Above(\vec{c}_r))$.*

Example 5.9 *Following Example 5.8, suppose we enlarge the source $Above(< a^2, b^1, c^1, d^1 >)$ by redrawing the upper curve, such that the cuboid $< a^1, b^2, c^2, d^2 >$ is now above the new curve. Then we have that $< a^1, b^2, c^2, d^2 > \wedge < a^2, b^1, c^1, d^1 > = < a^1, b^1, c^1, d^1 >$, indicating potential m-d inferences from the former two cuboids to the latter.*

As another aspect of the maximality, $Above(\vec{c}_r)$ is not arbitrary, but instead the only choice to satisfy both the prevention of m-d inferences and the maximality. This is formalized as the only if part in Theorem 1 (the if part is given by Lemma 1 and 2). Hence, $Above(\vec{c}_r)$ is the best possible result with respect to the prevention of m-d inferences. Any further improvement of this result will require the detection of m-d inferences, which is possible in only a few special cases, as discussed in Section 5.1.

Theorem 1 *In any data cube $\mathcal{L}$, given any l-specification S_l, any set of unprotected cuboids $C \subseteq \mathcal{L} \setminus Below(S_l)$ satisfies the following two properties, if and only if $C = Above(\vec{c}_r)$, where $\vec{c}_r$ is a minimal element of $\mathcal{L} \setminus Below(S_l)$.*

1. *$\mid Basis(C, \vec{c}) \mid = 1$, for any $\vec{c} \in \mathcal{L} \setminus C$.*
2. *$\mid Basis(S, \vec{c}) \mid > 1$, for any S satisfying $C \subset S \subseteq \mathcal{L} \setminus Below(S_l)$, and some $\vec{c} \in Below(S_l)$.*

5.2.3. The General Case Next we consider the general case of an authorization object $Object(O)$, where $O = \{(r_i, S_i) : 1 \leq i \leq n\}$ are pairs of s-specification and l-specification. As in Section 4, the key issue is the potentially exponential number of intersections among slices. Inference control will be prohibitive if it needs to deal with each such intersection.

The result stated in Lemma 1 is distributive over the set intersection operation. Without loss of generality, suppose neither $Below(S_1) \subseteq Below(S_2)$ nor $Below(S_2) \subseteq Below(S_1)$ holds, and $Object((r_1, S_1)) \cap Object((r_2, S_2)) \neq \phi$ (that is, some protected cells are included by the intersection). We divide $Slice(r_1)$ and $Slice(r_2)$ into three disjoint parts $Slice(r_1) \setminus Slice(r_2)$, $Slice(r_2) \setminus Slice(r_1)$ and $Slice(r_1) \cap Slice(r_2)$. Then by the third condition of Definiton 4, we can consider each part independently for inferences. By Theorem 1, we can prevent m-d inferences in $Slice(r_1) \setminus Slice(r_2)$ and $Slice(r_2) \setminus Slice(r_1)$ by choosing a root for each part respectively. Now for the intersection $Slice(r_1) \cap Slice(r_2)$, we simply regard the joint of the two chosen roots as a new root. Lemma 3 then states two results. Firstly, this new root is unprotected with respect to the closure $Below(S_1 \cup S_2)$. This means the root is indeed valid for preventing m-d inferences according to Lemma 1. Secondly, the function $Above()$ is *distributive* over set intersection. Hence, the intersection can be ignored in inference control, if the open policy is properly enforced. That is, the cells in the intersection are free of m-d inferences only if both slices say so.

Lemma 3 *In any data cube $\mathcal{L}$, given any two pair of s-specification and l-specification (r_1, S_1) and (r_2, S_2), let $\vec{c}_1 \in \mathcal{L} \setminus Below(S_1)$ and $\vec{c}_2 \in \mathcal{L} \setminus Below(S_2)$. Then both $\vec{c}_1 \vee \vec{c}_2 \in \mathcal{L} \setminus Below(S_1 \cup S_2)$ and $Above(\vec{c}_1 \vee \vec{c}_2) = Above(\vec{c}_1) \cap Above(\vec{c}_2)$ hold.*

However, while intersections are free of m-d inferences, the maximality stated in Lemma 2 is not guaranteed for them. When the two roots $\vec{c}_1$ and $\vec{c}_2$ are minimal elements of $\mathcal{L} \setminus Below(S_1)$ and $\mathcal{L} \setminus Below(S_2)$, respectively, their joint is not necessarily a minimal element of $\mathcal{L} \setminus Below(S_1 \cup S_2)$. In order to preserve the maximality for the intersection, we must coordinate the choice of both roots. In general, preserving maximality for all the intersections is difficult. Even a restricted case where the number of intersections is linear in that of slices is no easier than the *maximal independent set* problem, which is known as NP-complete [16]. Considering that intersections are usually not intentional, and the choices of roots also depend on other probably more important factors (such as the elimination of 1-d inferences discussed in Section 5.3), the maximality is not further pursued for intersections. We define the answerable set given by a set of roots in Definition 7. We also extend the function $Basis()$ defined in Definition 5.

Definition 7 *In any data cube $\mathcal{L}$, let $\mathcal{A}$ be the set of all cells,*

1. *define a function $Answerable(.) : 2^{2^{\mathcal{A}} \times \mathcal{L}} \to 2^{\mathcal{A}}$ as: $Answerable(\{(r_i, \vec{c}_i) : 1 \leq i \leq n\}) = \{\vec{t} :$ for $1 \leq i \leq n, \vec{t} \in Slice(r_i)$ implies $\vec{t} \in \vec{c}$ holds for some $\vec{c} \in Above(\vec{c}_i)\}$. We say $Answerable(R)$ is the* **answerable set** *given by the set of roots $R = \{(r_i, \vec{c}_i) : 1 \leq$*

$i \leq n\}$. We say each cell $\vec{t} \in Answerable(R)$ is **answerable***, and each cell $\vec{t} \in \mathcal{A} \setminus Answerable(R)$* **restricted***, and*

2. *define a function $gBasis(.) : 2^{\mathcal{A}} \times \mathcal{A} \rightarrow 2^{\mathcal{A}}$ as $gBasis(S, \vec{t_t}) = \{\vec{t} : \vec{t} \in S, \vec{t_t} \preceq \vec{t}, (\vec{t_1} \in S \wedge \vec{t_1} \preceq \vec{t})$ implies $\vec{t_1} = \vec{t}\}$. We say $gBasis(S, \vec{t_t})$ is the* **basis** *of a set of cells S with respect to a cell $\vec{t_t}$.*

The answerable set is characterized in Theorem 2. The first claim says that an answerable set given by unprotected roots is always free of m-d inferences to any restricted cell. The second claim shows that an answerable set is maximal if all the roots are minimal and no protected cells are included by intersections of slices. The third claim says that finding an appropriate set of roots to maximize the answerable set is NP-hard, if such disjoint slices are not the case. Notice that this NP-hardness result only states the infeasibility of maximizing the answerable set, but does not imply that of finding an answerable set.

Theorem 2 *In any data cube $\mathcal{L}$ with the set of all cells $\mathcal{A}$, given pairs of s-specification and l-specification $O = \{(r_i, S_i) : 1 \leq i \leq n)\}$, and a set of roots $R = \{(r_i, \vec{c_i}) : 1 \leq i \leq n, \vec{c_i}$ is a minimal element of $\mathcal{L} \setminus Below(S_i)\}$. The following conditions 1 and 2 hold, if $Object((r_i, S_i)) \cap Object((r_j, S_j)) = \phi$ is true for any $1 \leq i \neq j \leq n$; the following conditions 1 and 3 hold, otherwise.*

1. *$\mid gBasis(Answerable(R), \vec{t_t}) \mid = 1$, for any restricted cell $\vec{t_t} \in \mathcal{A} \setminus Answerable(R)$.*
2. *$\mid gBasis(S, \vec{t_t}) \mid > 1$, for any set of cells S satisfying $Answerable(R) \subset S \subseteq \mathcal{A} \setminus Object(O)$, and some protected cell $\vec{t_t} \in Object(O)$.*
3. *Finding the set R satisfying the above two conditions is NP-hard.*

5.3. Eliminating One-Dimensional Inferences

In this section, we address the elimination of 1-d inferences. For a given set of specifications $O = \{(r_i, S_i) : 1 \leq i \leq m\}$, the answerable set $Answerable(R)$ given by a set of unprotected roots $R = \{(r_i, \vec{c_i}) : 1 \leq i \leq n\}$ is free of m-d inferences. However, $Answerable(R)$ may not be truly answerable, because some of its cells may still cause 1-d inferences, as illustrated in Example 5.1. Without loss of generality, consider the first slice $Slice(r_1)$. Let $r \subseteq Answerable((r_1, \vec{c_1}))$ be the set of cells in $Slice(r_1)$ that cause 1-d inferences according to a given sensitivity criterion $Sensitive()$. It may seem a viable solution to simply restrict r to remove 1-d inferences from $Slice(r_1)$. However, such basic access control never works. Both m-d inferences and 1-d inferences are possible from the unrestricted cells in $Answerable((r_1, \vec{c_1})) \setminus r$ to those in r (and then to the protected cells in $Slice(r_1) \cap Object(O)$). On the other hand, considering r as a new object, then nothing prevents it from being protected by the same methods used to protect $Object(O)$. Specifically, by the third condition of Definition 4, we can consider $Slice(r_1) \setminus Slice(r)$ and $Slice(r)$ separately for inferences. The former is now free of both m-d inferences and 1-d inferences. Let S be the union of S_1 and the set of cuboids with which r has a non-empty intersection. Then $Object((r, S))$ is a new object to be protected, if we consider r and S as the s-specification and l-specification, respectively. Now we have reached the same point as to protect $Object(O)$. That is, we first find a new root $\vec{c_r}$ to prevent m-d inferences from $Answerable((r_1, \vec{c_1})) \setminus Object((r, S))$ to $Object((r, S))$. Then again, the result $Answerable((r, \vec{c_r}))$ is free of m-d inferences to $Object((r, S))$, but may cause 1-d inferences. Hence, we repeat the above process until either no more 1-d inferences are detected, or there is no cuboid left to be chosen for the root. Then the root is set as a dummy cuboid $\vec{c}_{nil}$, which depends on the topmost cuboid $< all, all, \ldots, all >$ (hence $Above(\vec{c}_{nil}) = \phi$, causing no 1-d inferences). The process terminates in at most $\mid Above(\vec{c_1}) \mid$ steps, because in each round the root is chosen from fewer candidates, with the cuboids causing 1-d inferences excluded. The procedure *SeCube* shown in Figure 3 computes a set of roots as described above.

Proposition 2 states that the result of the procedure *SeCube* given in Figure 3 is provably secure. The first claim says the set of restricted cells is itself an object, and this new object is a superset of the originally specified object (that is, the set of protected cells). This result has two implications. Firstly, the protection provided by inference control actually meets the requirements of the basic access control discussed at the end of Section 4 (notice that this is different from access control in general, which may go beyond the open policy we have assumed). Hence, unauthorized accesses to the specified object are eliminated by the inference control, if the authorizations are specified as in Section 4. Intuitively, any protected cells (explicitly specified so by administrators) will be restricted, although the reverse may not be true. Secondly, the result of such inference control can be enforced through access control, since it is nothing but an object. The task of implementing inference control can thus be fulfilled using existing access control mechanisms. The second claim in Proposition 2 states that the answerable set is always free of inferences to restricted cells, regardless what sensitivity criterion is given, as long as it meets the requirements stated in Definition 4. This claim implies that sensitivity criteria can be implemented as customizable modules, in order to incorporate the different requirements in different applications.

Proposition 2 *Taken as input a data cube $\mathcal{L}$ with the set of all cells $\mathcal{A}$, pairs of s-specification and l-specification O, and any sensitivity criterion $Sensitive()$, the procedure* Se-

```
Procedure SeCube
Input: a data cube $\mathcal{L}$ with the set of all cells $\mathcal{A}$, pairs of s-specification and l-specification $\{(r_i, S_i) : 1 \leq i \leq n\}$,
        and a sensitivity criterion $Sensitive()$
Output: $R = \{(u_i, \vec{w}_i) : 1 \leq i \leq l\}$, where each $u_i$ is an s-specification, and $\vec{w}_i$ is a root
Method:
    Let $\mathcal{L} = \mathcal{L} \cup \{\vec{c}_{nil}\}$, where $\vec{c}_{nil}$ satisfies $< all, all, \ldots, all > \preceq \vec{c}_{nil}$, and $R = \phi$
    For $i = 1$ to $n$
        Let $R = R \cup \{(r_i, \vec{c}_i)\}$, where $\vec{c}_i \in \mathcal{L} \setminus Below(S_i)$
        Let $r = r_i$, $S = S_i$, and $\vec{c}_r = \vec{c}_i$
        While $TRUE$
            Let $Ans = Answerable((r, \vec{c}_r))$ and $r = \{\vec{t} : \vec{t} \in Ans, Sensitive(\vec{t}, \mathcal{A} \setminus Ans) = TRUE\}$
            If   $r = \phi$ Break
            Let $S = S \cup \{\vec{c} : \vec{c} \in Above(\vec{c}_r), \vec{c} \cap r \neq \phi\}$
            Let $\vec{c}_r \in Above(\vec{c}_r) \setminus Below(S)$
            Let $R = R \cup \{(r, \vec{c}_r)\}$
    Return $R$
```

Figure 3. A Procedure for Inference Control in Data Cubes

Cube *in Figure 3 outputs a set* $R = \{(u_i, \vec{w}_i) : 1 \leq i \leq l\}$ *satisfying*

1. $\mathcal{A} \setminus Answerable(R) = Object(O_a)$ *and* $Object(O_a) \supseteq Object(O)$ *both hold, where* $O_a = \{(u_i, W_i) : 1 \leq i \leq l\}$ *and each* W_i *is the set of maximal elements of* $\mathcal{L} \setminus Above(\vec{w}_i)$*, and*
2. $Sensitive(Answerable(R), Object(O_a)) = FALSE$ *is true.*

6. Implementation Options and Complexity

The methods proposed in previous sections provably eliminate both unauthorized accesses and malicious inferences in data cubes. In this section, we discuss the implementation options and the computational and storage requirements.

The implementation options largely depend on the existing security capability of an OLAP system. Most ROLAP systems have access control mechanisms in their relational backend. Such mechanisms can check incoming queries and deny any request for protected cells. However, the access control mechanisms are not aware of inferences. Our methods can enhance those existing mechanisms with the capability of inference control. Proposition 2 states that the result of inference control is simply a *larger* object containing restricted cells, which encloses the originally specified object containing protected cells. Once this result of inference control is computed off-line by the Procedure *SeCube*, it can be enforced by existing access control mechanisms, in the same way as the specified object is enforced. That is, the request for any restricted cell will be denied. Ideally, such an *inference control-through access control* approach brings no additional on-line performance overhead to the OLAP system with access control already in place. Because a query needs to be checked by access control mechanisms anyway. The distributivity stated in Proposition 1 and Lemma 3 also implies that the result of inference control can be incrementally maintained, when new specifications arrive. Hence, old specifications do not need to be kept once the result is computed, meaning no additional storage is required, either.

For an OLAP system with no existing access control mechanisms, the implementation can adopt a *query-modification* approach. The core cuboid $\vec{c}_c$ is materialized in most data cubes [19]. Hence, the answerable set can be encoded by attaching the root to each cell in $\vec{c}_c$. The root can be implemented as a new attribute of the size $O(k)$, for a k-dimensional data cube. The storage requirement is thus $O(\mid \vec{c}_c \mid \cdot k)$. After a request for a set of cells S in a cuboid $\vec{c}$ is received, the system modifies the request by appending to it a simple restriction. The restriction says that a cell in the core cuboid $\vec{c}_c$ is aggregated by the answer only if its root is an ancestor of the requested cuboid $\vec{c}$ (in ROLAP this will be a simple *WHERE* clause, which is appended to the SQL command formed by the front-end). Such a restriction ensures that all aggregated cells are included by the answerable set, and hence lead to neither inferences nor unauthorized accesses. The dependency relationship between any two cuboids can be checked in $O(k)$ time (by comparing two k-tuples), and hence the on-line delay is $O(\mid S \mid \cdot k)$. Additional processing may be necessary to inform the subject about the restricted cells. This approach causes on-line performance delay and also requires modifications of existing OLAP systems. However, the cost of our solution is comparable to that of the basic access control (that is, denying any request for protected cells).

The off-line complexity of our solution depends on the implementation and optimization of the procedure *SeCube* shown in Figure 3. In the worst case, the complexity of *SeCube* is bounded by the number of specifications, the size of the data cube, and the size of the dependency lattice. The outer loop of the procedure runs n times, where n is the number of specification pairs. The inner loop of the procedure runs at most $\mid Above(\vec{c}_i) \mid$ times for the i^{th} round of the outer loop. 1-d inferences are detected once for each protected cell $\vec{t}_t$ (against its single descendant in the basis $gBasis(Answerable(R), \vec{t}_t)$, according to Theorem 2) in the slice $Slice(r_i)$, whose results are stored and reused. The total number of detections is thus bounded by $O(\mid \mathcal{A} \mid)$. The detection is done by a customizable module that implements the function $Sensitive()$ in Definition 4. Each such detection usually takes constant time (such as checking whether a requested cell has less than two ancestors in a specific cuboid). In practice, the likelihood of 1-d inferences can usually be estimated based on statistics such as cuboid sizes [11]. Hence, the choice of roots can be optimized to reduce both the number of 1-d inferences in the initial answerable set, and the number of rounds in eliminating 1-d inferences. In most OLAP systems, materializations and pre-computations are extensively adopted, and off-line computational complexity is usually not a major concern in those systems. Our approach also reflects such an effort in achieving less on-line delay through more off-line processing. Our ongoing work addresses the evaluation and optimization of the proposed methods.

7. Conclusion

We provided a solution to protecting sensitive data in OLAP data cubes from unauthorized accesses and malicious inferences. We first devised a flexible framework for specifying authorization objects. We then proposed a method for controlling malicious inferences. The result is provably secure in that both malicious inferences and unauthorized accesses are eliminated. The method is applicable to any aggregation functions, external knowledge, and sensitivity criteria, given that they satisfy the stated properties. The technique is efficient and its implementation requires little modifications to existing OLAP systems. As the first part of a continuing effort to secure OLAP systems, this paper has outlined our approach and especially focused on the security aspect. Ongoing and future work addresses other important aspects, including the tradeoff between the efficiency and usefulness of a secure OLAP system, the optimal choices of roots, and the variation in which, instead of being static, the roots are driven down by queries.

Acknowledgements The authors are grateful to the anonymous reviewers for their valuable comments.

References

[1] N.R. Adam and J.C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the Nineteenth ACM SIGMOD Conference on Management of Data (SIGMOD'00)*, pages 439–450, 2000.

[3] L.L. Beck. A security mechanism for statistical databases. *ACM Trans. on Database Systems*, 5(3):316–338, 1980.

[4] B. Bhargava. Security in data warehousing (invited talk). In *Proceedings of the 3rd Data Warehousing and Knowledge Discovery (DaWak'00)*, 2000.

[5] L. Brankovic, M. Miller, P. Horak, and G. Wrightson. Usability of compromise-free statistical databases. In *Proceedings of the Ninth International Conference on Scientific and Statistical Database Management (SSDBM '97)*, pages 144–154, 1997.

[6] F.Y. Chin. Security problems on inference control for sum, max, and min queries. *Journal of the Association for Computing Machinery*, 33(3):451–464, 1986.

[7] F.Y. Chin and G. Özsoyoglu. Statistical database design. *ACM Trans. on Database Systems*, 6(1):113–139, 1981.

[8] F.Y. Chin and G. Özsoyoglu. Auditing and inference control in statistical databases. *IEEE Trans. on Software Engineering*, 8(6):574–582, 1982.

[9] L.H. Cox. Suppression methodology and statistical disclosure control. *Journal of American Statistical Association*, 75(370):377–385, 1980.

[10] L.H. Cox. On properties of multi-dimensional statistical tables. *Journal of Statistical Planning and Inference*, 117(2):251–273, 2003.

[11] D.E. Denning. *Cryptography and data security*. Addison-Wesley, Reading, Massachusetts, 1982.

[12] D.E. Denning and J. Schlörer. Inference controls for statistical databases. *IEEE Computer*, 16(7):69–82, 1983.

[13] D. Dobkin, A.K. Jones, and R.J. Lipton. Secure databases: protection against user influence. *ACM Trans. on Database Systems*, 4(1):97–106, 1979.

[14] T. Donnellan. *Lattice Theory*. Pergamon press, Oxford, 1968.

[15] L.P. Fellegi and A.B. Sunter. A theory for record linkage. *Journal of American Statistic Association*, 64(328):1183–1210, 1969.

[16] M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[17] J. Gray, A. Bosworth, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, crosstab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[18] P. Griffiths and B.W. Wade. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems*, 1(3):242–255, September 1976.

[19] V. Harinarayan, A. Rajaraman, and J.D. Ullman. Implementing data cubes efficiently. In *Proceedings of the Fifteenth ACM SIGMOD international conference on Management of data (SIGMOD'96)*, pages 205–227, 1996.

[20] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(4):1–57, dec 2001.

[21] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *Proceedings of the Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'00)*, pages 86–91, 2000.

[22] Y. Li, L. Wang, X.S. Wang, and S. Jajodia. Auditing interval-based inference. In *Proceedings of the Fourteenth Conference on Advanced Information Systems Engineering (CAiSE'02)*, pages 553–568, 2002.

[23] B. Malin, L. Sweeney, and E. Newton. Trail re-identification: Learning who you are from where you have been. Technical Report, 2003. Available at http://privacy.cs.cmu.edu/.

[24] J.M. Mateo-Sanz and J. Domingo-Ferrer. A method for data-oriented multivariate microaggregation. In *Proceedings of the Conference on Statistical Data Protection'98*, pages 89–99, 1998.

[25] H.B. Newcombe, J.M. Kennedy, S.J. Axford, and A.P. James. Automatic linkage of vital records. *Science*, 130(3381):954–959, 1959.

[26] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[27] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[28] J. Schlörer. Security of statistical databases: multidimensional transformation. *ACM Trans. on Database Systems*, 6(1):95–112, 1981.

[29] A. Shoshani. OLAP and statistical databases: Similarities and differences. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, pages 185–196, 1997.

[30] G. Pernul T. Priebe. Towards olap security design - survey and research issues. In *Proceedings of 3rd ACM International Workshop on Data Warehousing and OLAP (DOLAP'00)*, pages 114–121, 2000.

[31] J.F. Traub, Y. Yemini, and H. Woźniakowski. The statistical security of a statistical database. *ACM Trans. on Database Systems*, 9(4):672–679, 1984.

[32] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, pages 639–644, 2002.

[33] L. Wang, Y.J. Li, D. Wijesekera, and S. Jajodia. Precisely answering multi-dimensional range queries without privacy breaches. In *Proceedings of the Eighth European Symposium on Research in Computer Security (ESORICS'03)*, 2003.

[34] L. Wang, D. Wijesekera, and S. Jajodia. Cardinality-based inference control in sum-only data cubes. In *Proceedings of the Seventh European Symposium on Research in Computer Security (ESORICS'02)*, pages 55–71, 2002.

[35] L. Willenborg and T. de Walal. *Statistical disclosure control in practice*. Springer Verlag, New York, 1996.

[36] C.T. Yu and F.Y. Chin. A study on the protection of statistical data bases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'77)*, pages 169–181, 1977.

Appendix

Proof of (Proposition 1) We show that $Below(S_1 \cup S_2) \subseteq Below(S_1) \cup Below(S_2)$ and $Below(S_1) \cup Below(S_2) \subseteq Below(S_1 \cup S_2)$ hold. By Definition 1, $Below(S_1 \cup S_2) = \{\vec{c} : \exists \vec{c}_s \in S_1 \cup S_2\ \vec{c} \preceq \vec{c}_s\}$. Hence, for any $\vec{c} \in Below(S_1 \cup S_2)$, either $\vec{c} \succeq \vec{c}_1 \in S_1$ or $\vec{c} \succeq \vec{c}_2 \in S_s$ (or both) holds. That is, $Below(S_1 \cup S_2) \subseteq Below(S_1) \cup Below(S_2)$ holds. Conversely, any $\vec{c}_1 \in Below(S_1)$ must satisfy $\vec{c}_1 \succeq \vec{c} \in S_1 \subseteq S_1 \cup S_2$, and hence $\vec{c}_1 \in Below(S_1 \cup S_2)$ holds. Similar case for any $\vec{c}_2 \in Below(S_2)$. Hence, $Below(S_1) \cup Below(S_2) \subseteq Below(S_1 \cup S_2)$ holds. This shows that $Below(S_1) \cup Below(S_2) = Below(S_1 \cup S_2)$ is true. □

Proof of (Lemma 1) By Definition 5, we have that any $\vec{c} \in Basis(Above(\vec{c}_r), \{\vec{c}_t\})$ only if $\vec{c} \in Above(\vec{c}_r)$ and $\vec{c} \succeq \vec{c}_t$ both hold. Then by Definition 6, $\vec{c} \succeq \vec{c}_r$ must also be true. Hence, $Basis(Above(\vec{c}_r), \{\vec{c}_t\}) \subseteq S = \{\vec{c} : \vec{c} \succeq \vec{c}_r, \vec{c} \succeq \vec{c}_t\}$ holds. Clearly, $\vec{c}_r \vee \vec{c}_t \in S$ holds. Moreover, any $\vec{c} \in S$ must satisfy $\vec{c} \succeq \vec{c}_r \vee \vec{c}_t$. Then again by By Definition 5, any $\vec{c} \succ \vec{c}_r \vee \vec{c}_t$ must not satisfy $\vec{c} \in Basis(Above(\vec{c}_r), \{\vec{c}_t\})$, because otherwise $\vec{c} \succ \vec{c}_r \vee \vec{c}_t$ implies $\vec{c} = \vec{c}_r \vee \vec{c}_t$, a contradiction. Consequently, $Basis(Above(\vec{c}_r), \{\vec{c}_t\}) \subseteq \{\vec{c}_r \vee \vec{c}_t\}$ is true. On the other hand, the definition of joint says that $\vec{c} \in S$ and $\vec{c} \preceq \vec{c}_r \vee \vec{c}_t$ implies $\vec{c} = \vec{c}_r \vee \vec{c}_t$. That is, $\vec{c}_r \vee \vec{c}_t \in Basis(Above(\vec{c}_r), \{\vec{c}_t\})$ is also true. This implies $Basis(Above(\vec{c}_r), \{\vec{c}_t\}) = \{\vec{c}_r \vee \vec{c}_t\}$. □

Proof of (Lemma 2) Because $\vec{c}_r$ is a minimal element of $\mathcal{L} \setminus Below(S_l)$, $\vec{c}_r \wedge \vec{c}_1 \preceq \vec{c}_r$ implies either $\vec{c}_r \wedge \vec{c}_1 = \vec{c}_r$ or $\vec{c}_r \wedge \vec{c}_1 \notin \mathcal{L} \setminus Below(S_l)$ holds. However, $\vec{c}_r \wedge \vec{c}_1 = \vec{c}_r$ must not hold, because otherwise $\vec{c}_1 \succeq \vec{c}_r$ implies the contradiction $\vec{c}_1 \in Above(\vec{c}_r)$. That is, $\vec{c}_r \wedge \vec{c}_1 \in Below(S_l)$ holds. □

Proof of (Theorem 1) The if part is given by Lemma 1 and Lemma 2. We only show the only if part. Any non-empty $C \subseteq \mathcal{L} \setminus Below(S_l)$ is a poset , and hence has a unique non-empty set of minimal elements C_m. We first show that $\mid C_m \mid > 1$ must not hold by contradiction. Let $\vec{c}$ be GLB of C_m. Then $\vec{c} \in Below(S_l)$ must not hold. Otherwise, by Definition 5, $Basis(C, \vec{c}) = C_m$ and hence $\mid Basis(C, \vec{c}) \mid = \mid C_m \mid > 1$ is a contradiction to the first condition. Hence, $\vec{c} \in \mathcal{L} \setminus Below(S_l)$ is true. Because $\vec{c}$ is the GLB of C_m, it is also the GLB of C. Then by Definition 6, $C \subseteq Above(\vec{c})$ holds. However, $\mid C_m \mid > 1$ implies $C = Above(\vec{c})$ must not hold, because $Above(\vec{c})$ has only one minimal element $\vec{c}$. Hence, $C \subset Above(\vec{c})$. This contradicts the second condition of maximality, because $Above(\vec{c})$ clearly satisfies the first condition. Consequently, $\mid C_m \mid = 1$ must hold. Let $C_m = \{\vec{c}\}$. Then again, to satisfy the maximality, $C = Above(\vec{c})$ must be true. □

Proof of (Lemma 3) We first show $\vec{c}_1 \vee \vec{c}_2 \in Below(S_1 \cup S_2)$ must not hold by contradiction. Suppose it is true, then by Proposition 1, either $\vec{c}_1 \vee \vec{c}_2 \in Below(S_1)$ or $\vec{c}_1 \vee \vec{c}_2 \in Below(S_1)$ (or both) holds. However, if $\vec{c}_1 \vee \vec{c}_2 \in Below(S_1)$ is true, then $\vec{c}_1 \preceq \vec{c}_1 \vee \vec{c}_2 \preceq \vec{c}$ must hold for some $\vec{c} \in S_1$. This leads to the contradiction $\vec{c}_1 \in Below(S_1)$. Similarly, $\vec{c}_1 \vee \vec{c}_2 \in Below(S_1)$ implies $\vec{c}_2 \in Below(S_2)$. This shows that $\vec{c}_1 \vee \vec{c}_2 \in \mathcal{L} \setminus Below(S_1 \cup S_2)$ must be true. Then we show $Above(\vec{c}_1 \vee \vec{c}_2) = Above(\vec{c}_1) \cap Above(\vec{c}_2)$ also holds. By Definition 6, $\vec{c} \in Above(\vec{c}_1 \vee \vec{c}_2)$ holds if and only if $\vec{c} \succeq \vec{c}_1 \vee \vec{c}_2$ and hence both $\vec{c} \succeq \vec{c}_1$ and $\vec{c} \succeq \vec{c}_1$ hold. This is equivalent to saying $\vec{c} \in Above(\vec{c}_1) \cap Above(\vec{c}_2)$. □

Proof of (Theorem 2) We first justify the first claim. Suppose $\vec{t_t} \in \vec{c_t}$ holds for some $\vec{c_t} \in \mathcal{L}$, and let $I = \{i : 1 \leq i \leq n, \vec{t_t} \in Slice(r_i)\}$. Then $\vec{t_t} \in \mathcal{A} \setminus Answerable(R)$ implies that $\vec{c_t} \in \bigcup_{i\in I} Below(S_i) = Below(\bigcup_{i\in I} S_i)$ holds. By Definition 7, $gBasis(Answerable(R), \vec{t_t}) \subseteq Answerable(R) \cap (\bigcup_{i\in I} Slice(r_i)) = (\bigcap_{i\in I} Above(\vec{c_i})) \cap (\bigcup_{i\in I} Slice(r_i))$. By Lemma 3, we have that $gBasis(Answerable(R), \vec{t_t}) \subseteq Above(\bigvee_{i\in I} \vec{c_i}) \cap (\bigcup_{i\in I} Slice(r_i))$ and $\bigvee_{i\in I} \vec{c_i} \in \mathcal{L} \setminus Below(\bigcup_{i\in I} S_i)$, where $\bigvee_{i\in I} \vec{c_i}$ dentoes the LUB of $\{c_i : i \in I\}$. By Lemma 1, $Basis(Above(\bigvee_{i\in I} \vec{c_i}), \vec{c_t}) = \{\vec{c_t} \cap \bigvee_{i\in I} \vec{c_i}\}$ holds. Denotes $\vec{c_t} \cap \bigvee_{i\in I} \vec{c_i}$ by $\vec{c_r}$. Because $\vec{c_t} \preceq \vec{c_r}$, there must exist one and only one $\vec{t_r} \in \vec{c_r}$ satisfying $\vec{t_t} \preceq \vec{t_r}$. Then $\vec{t_r} \in S$ holds, because both $\vec{t_r} \in \vec{c_r} \in Above(\bigvee_{i\in I} \vec{c_i})$ and $\vec{t_r} \in \bigcup_{i\in I} Slice(r_i)$ are true. For any $\vec{t} \in S$ satisfying $\vec{t_t} \preceq \vec{t}$, if $\vec{t} \in \vec{c}$ then $\vec{c_r} \preceq \vec{c}$ must hold, and $\vec{t_r} \preceq \vec{t}$ follows. Hence, $\vec{t} \preceq \vec{t_r}$ will imply $\vec{t} = \vec{t_r}$. That is, $gBasis(Answerable(R), \vec{t_t}) = \{\vec{t_r}\}$.

Next we justify the second claim by contradiction. Suppose some S satisfies both $Answerable(R) \subset S \subseteq \mathcal{A} \setminus Object(O)$ and $\mid gBasis(S, \vec{t_t}) \mid = 1$ for any $\vec{t_t} \in Object(O)$. Let $\vec{t_s} \in S \setminus Answerable(R)$, and $\vec{t_s} \in \vec{c_s}$ for some $\vec{c_s} \in \mathcal{L}$. Because $Object(\{(r_i, S_i)\}) \cap Object(\{(r_j, S_j)\}) = \phi$ is true for any $1 \leq i \neq j \leq n$, $\mid I = \{i : 1 \leq i \leq n, \vec{t_s} \in Slice(r_i)\} \mid = 1$ must be true. Without loss of generality, suppose $\vec{t_s} \in Slice(r_1)$ holds. Then $\vec{t_s} \in S \subseteq \mathcal{A} \setminus Object(O)$ implies $\vec{c_s} \in \mathcal{L} \setminus Below(S_1)$. Moreover, $\vec{t_s} \in Slice(r_1)$ and $\vec{t_s} \notin Answerable(R)$ implies $\vec{c_s} \notin Above(\vec{c_1})$. By Lemma 2, $\vec{c_s} \wedge \vec{c_1} \in Below(S_1)$. Denote $\vec{c_s} \wedge \vec{c_1}$ as $\vec{t_t}$, and suppose $\vec{t_t} \in \vec{c_t} \in Below(S_1)$. Then $\vec{t_t} \in Object(O)$ follows. Because $\vec{c_t} \preceq \vec{c_1}$, there must exist $\vec{t_1} \in \vec{c_1} \cap Slice(r_1) \subseteq Answerable(R)$ satisfying $\vec{t_t} \preceq \vec{t_1}$. Moreover, $\vec{c_s} \notin Above(\vec{c_1})$ implies $\vec{c_s}$ and $\vec{c_1}$ are non-comparable, and hence $\vec{t_1}$ and $\vec{t_s}$ are also non-comparable. Consequently, $gBasis(S, \vec{t_t}) \supseteq \{\vec{t_s}, \vec{t_1}\}$, a contradiction to $\mid gBasis(S, \vec{t_t}) \mid = 1$.

Finally we show the third claim holds by reducing the *maximal independent set* problem to a restricted case of our problem. Given any simple undirected graph $G(V, E)$, with the vertex set $V = \{i : 1 \leq i \leq n\}$, the edge set $E = \{\{i, j\} : 1 \leq i \neq j \leq n\}$, and the vertices i and j being incidenced by the edge $\{i, j\}$, we construct a new graph $G'(V', E')$ as the follows. First we let $G' = G$. Then for any isolated vertex $i \in V$, we add a new vertex $2i$ to V' and a new edge $\{i, 2i\}$ to E'. Then for any vertex $i \in V$ that is incidenced by only one edge, we add a new vertex $3i$ to V' and a new edge $\{i, 3i\}$ to E'. The new graph G' satisfies that $\mid V' \mid \leq 3n$, and each $1 \leq i \leq n$ in V' is incidenced by at least two edges, while each vertex $i > n$ in V' is incidenced by a single edge. For each vertex $1 \leq i \leq n$ in V', we use I_i for the set of its adjacent vertices $\{j : \{i, j\} \in E'\}$.
We then construct a data cube $\mathcal{L}$, so a subset $S = \{\vec{c_i} : 1 \leq i \leq 3n\} \subseteq \mathcal{L}$ satisfies that any two cuboids in S are non-comparable, and their joint is the topmost cuobid $\vec{c}_{all} =< all, all, \ldots, all >$. For any edge $\{i, j\} \in E$, we let S_{ij} be the maximal elements of the set $\mathcal{L} \setminus \{\vec{c_i}, \vec{c_j}, \vec{c}_{all}\}$. Considering S_{ij} as an l-specification, then $\vec{c_i}$ and $\vec{c_j}$ are the two minimal elements of $\mathcal{L} \setminus Below(S_{ij})$, while $\vec{c_h} \in Below(S_{ij})$ holds for any $h \neq i$ and $h \neq j$. For each S_{ij}, we also choose one s-specification r_{ij}, such that for any $1 \leq i \leq n$ and any $J \subseteq I_i$, $\bigcap_{j\in J} Object(\{(r_{ij}, S_{ij}\}) \neq \phi$ holds, while $\bigcap_{j\in J\cup\{h\}} Object(\{(r_{ij}, S_{ij}\}) = \phi$ holds for any $h \notin I_i$. That is, any two objects overlap if and only if the two corresponding edges incidence the same vertex. Let $O = \{(r_{ij}, S_{ij}) : \{i, j\} \in E\}$. Let $R = \{(r_{ij}, \vec{c}_{ij}) : \{i, j\} \in E, \vec{c}_{ij} \in \{\vec{c_i}, \vec{c_j}\}\}$.

We claim that if we can find an R such that $Answerable(R)$ is maximal, then we immediately have $V_s \subseteq V$ such that for any $i \in V_s$ and $j \in V_s$, $\{i, j\} \notin E$, and V_s is maximal. The latter is an instance of the maximal independent set problem, which is known as NP-complete [16], and hence finding an R to maximize $Answerable(R)$ is NP-hard. Now we justify the claim. Because for $1 \leq i \leq n$, each root $\vec{c}_{ij} \in \{\vec{c_i}, \vec{c_j}\}$ is a minimal element of $\mathcal{L} \setminus Below(S_{ij})$, $Answerable(R)$ is maximal if and only if the intersections are. Without loss of generality, consider the intersection at $\vec{c_1}$. That is, $(\bigcup_{i\in I_1} Slice(r_{1i})) \cap Answerable(R) = (\bigcup_{i\in I_1} Slice(r_{1i})) \cap Above(\bigvee_{i\in I_1} \vec{c}_{1i})$. $Above(\bigvee_{i\in I_1} \vec{c}_{1i})$ is maximal if and only if $\vec{c}_{1i} = \vec{c_1}$ for all $i \in I_1$, and hence $(\bigvee_{i\in I_1} \vec{c}_{1i}) = \vec{c_1}$. Because if $\vec{c}_{lj} = \vec{c_j}$ holds for some j, then $(\bigvee_{i\in I_1} \vec{c}_{1i}) = \vec{c_1} \wedge \vec{c_j} = \vec{c}_{all}$. We know that $Above(\vec{c}_{all})$ includes only a single cell, which must be less than the number of cells in $Above(\vec{c_1})$ (considering that $\mid \vec{c_i} \mid >> 1$ for any i). However, by assigning $\vec{c}_{1i} = \vec{c_1}$ for all $i \in I_1$, the answerable set of the intersection at $\vec{c_1}$ is maximized, but those of the intersections at $\vec{c_i}$ for $i \in I_1$ are not maximized. Consequently, $Answerable(R)$ is maximal if and only if for most of i $(1 \leq i \leq n)$, $\vec{c}_{ij} = \vec{c_i}$ holds for all $j \in I_i$. This is equivalent to finding a maximal subset $V_s \subseteq [1, n]$, so that for any $i \in V_s$ and $j \in V_s$, we can let $\vec{c}_{ih} = \vec{c_i}$ and $\vec{c}_{jk} = \vec{c_j}$ $(h \in I_i$ and $k \in I_j)$. This is possible only if $\{i, j\} \notin E$ holds for all $i \in V_s$ and $j \in V_s$. That is, V_s is the maximal independent set of $V = [1, n]$. □

Proof of (Proposition 2) We first justify the first claim. In order to show that $\mathcal{A} \setminus Answerable(R) \subseteq Object(O_a)$ holds, for any $\vec{t} \in \mathcal{A} \setminus Answerable(\{(u_i, \vec{w_i}) : 1 \leq i \leq l\})$, suppose $\vec{t} \in \vec{c_t}$ for some $\vec{c_t} \in \mathcal{L}$. Let $I = \{i : 1 \leq i \leq l, \vec{t} \in Slice(u_i)\}$. By Definition 7, $\vec{c_t} \notin Above(\vec{w_j})$ holds for some $j \in I$. Equivalently, $\vec{c_t} \in \mathcal{L} \setminus Above(\vec{w_j})$ holds. Then $\vec{c_t} \preceq \vec{c_w}$ holds for some $\vec{c_w} \in W_j$, and hence $\vec{c_t} \in Below(W_j)$ and $\vec{c_t} \in Below(\bigcup_{i\in I} W_i)$ both follow. Consequently, $\vec{t} \in (\bigcup_{i\in I} Slice(u_i)) \cap Below(\bigcup_{i\in I} W_i) \subseteq Object(O_a)$ is true. This shows $\mathcal{A} \setminus Answerable(R) \subseteq Object(O_a)$ is true. In order to show that $\mathcal{A} \setminus Answerable(R) \supseteq Object(O_a)$ also holds, for any $\vec{t} \in Object(O_a)$, suppose $\vec{t} \in \vec{c_t}$ for some $\vec{c_t} \in \mathcal{L}$. Let $I = \{i : 1 \leq i \leq l, \vec{t} \in Slice(u_i)\}$. Then $\vec{c_t} \in Below(\bigcup_{i\in I} W_i)$ implies that $\vec{c_t} \notin Above(W_j)$ must hold for some $j \in I$. Consequently, $\vec{c_t} \notin Answerable(R)$ is true and $\mathcal{A} \setminus Answerable(R) \supseteq Object(O_a)$ follows.

Next we show $Object(O_a) \supseteq Object(O)$ holds. For any $\vec{t} \in Object(O)$, suppose $\vec{t} \in \vec{c_t}$ for some $\vec{c_t} \in \mathcal{L}$. Let $I = \{i : 1 \leq i \leq n, \vec{t} \in Slice(r_i)\}$. By Definition 3, $\vec{c_t} \in Below(\bigcup_{i\in I} S_i)$ holds. According to the procedure *SeCube* in Figure 3, $(r_i, \vec{c_i}) \in R$ holds for all $1 \leq i \leq n$. Hence, $I \subseteq J = \{i : 1 \leq i \leq l, \vec{t} \in Slice(u_i)\}$ is true. Then $\vec{c_t} \in Below(\bigcup_{i\in I} W_i) \subseteq Below(\bigcup_{i\in J} W_i)$ also holds. Consequently, $\vec{t} \in (\bigcup_{i\in J} Slice(u_i)) \cap Below(\bigcup_{i\in J} W_i) \subseteq Object(O_a)$ is true.

Finally we justify the second claim by contradiction. Firstly, by Theorem 2, m-d inferences are impossible. That is, $Sensitive(Answerable(R), Object(O_a)) = TRUE$ only if for some $\vec{t} \in Answerable(R)$, $Sensitive(\vec{t}, Object(O_a)) = TRUE$ holds. Then according to the procedure *SeCube* in Figure 3, for some $1 \leq j \leq l$, $\vec{t} \notin Above(\vec{w_j})$ must hold. Hence, $\vec{t} \notin Answerable((u_j, \vec{w_j}))$ must be true, because $Answerable((u_j, \vec{w_j})) = Above(\vec{w_j}) \cap Slice(u_j)$. By Definition 7, $\vec{t} \in Answerable(R)$ cannot hold, either, which is a contradiction. □