Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Sciences 176 (2006) 1215–1240
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A cubic-wise balance approach for privacy preservation in data cubes

Yao Liu [a], Sam Y. Sung [a,*], Hui Xiong [b]

[a] *Department of Computer Science, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore*
[b] *Department of Computer Science, University of Minnesota—Twin Cities*

Received 5 October 2004; received in revised form 11 March 2005; accepted 14 March 2005

Abstract

A data warehouse stores current and historical records consolidated from multiple transactional systems. Securing data warehouses is of ever-increasing interest, especially considering areas where data are sold in pieces to third parties for data mining practices. In this case, existing data warehouse security techniques, such as data access control, may not be easy to enforce and can be ineffective. Instead, this paper proposes a data perturbation based approach, called the cubic-wise balance method, to provide privacy preserving range queries on data cubes in a data warehouse. This approach is motivated by the following observation: analysts are usually interested in summary data rather than individual data values. Indeed, our approach can provide a closely estimated summary data for range queries without providing access to actual individual data values. As demonstrated by our experimental results on APB benchmark data set from the OLAP council, the cubic-wise balance method can achieve both better privacy preservation and better range query accuracy than random data perturbation alternatives.

* Corresponding author. Tel.: +65 6874 6148.
E-mail address: ssung@comp.nus.edu.sg (S.Y. Sung).

0020-0255/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2005.03.010

Keywords: Data warehouse; OLAP; Privacy preservation; Range query; Data perturbation

1. Introduction

Data warehouse is built primarily as an open system to support On-line Analysis Processing (OLAP), which requires the open nature of data warehouse. However, with the growth of OLAP, the range of data warehouse users is growing steadily to include customers, partners or even third parties. This leads to privacy concern [1] and the needs of proper access control policies. Indeed, inappropriate disclosure of sensitive data stored in the underlying data warehouses may result in the breach of individual's privacy and jeopardize the organization's interest. It is well recognized that access control alone is insufficient in controlling information disclosure, because information not directly released may be inferred indirectly by manipulating legitimate queries about aggregated information.

Data cubes in a data warehouse are used to support data analysis. Specifically, a data cube is used to represent data along some measures of interest. In general, a data cube can be 2-dimensional, 3-dimensional, or higher dimensional. Each dimension represents an attribute in the data warehouse and the cells in the cube represent the measure of interest. Range query is one of the most important queries of data cube. Range queries apply a given aggregation operation over selected cells where the selection is specified as contiguous ranges in the domains of some of the attributes. Range-Sum query is finding the summation values over selected cells of a data cube where the selection is specified by a range of contiguous values for each dimension. For example, to find the total sales of stationary items, which have item codes ranging from 1201 to 1300, between day 130 and 159 in the western outlets, with branch-no ranging from 45 to 89 is a range-sum query.

Consider that a single data item in a data cube is not likely to be accessed alone, but a number of data are often aggregated to give summarized information and the trends of the database [2]. Preserving the confidential information in individual data cells while still being able to provide an accurate estimation of aggregations is the main focus of this paper. More specifically, our objectives are listed as follows:

(1) *Privacy preservation:* The individual cell data in a data cube is sensitive and must be protected from unauthorized users.
(2) *High accuracy:* Query results with high accuracy are valuable for business decision making.
(3) *Accessibility:* Most OLAP applications demand instant responses to interactive queries, although those queries usually aggregate a large

amount of data [3,4]. Hence, there should be no unnecessary restriction or data access control on the data.

To achieve the above three objectives, in this paper, we propose a cubic-wise balance data perturbation method to provide privacy preserving range queries on data cubes. This method is different from random data perturbation alternatives, since it provides a purposive perturbation on data cells in a way such that a closely estimated summary data for range queries can be obtained without providing access to the actual individual data values. As demonstrated by our experimental results on APB benchmark data set from the OLAP council, the cubic-wise balance method can achieve both better privacy preservation and better range query accuracy than random data perturbation alternatives.

Overview. The remainder of this paper is organized as follows: Section 2 presents related works. In Section 3, we introduce the cubic-wise balance method for privacy preserving range query in data cubes. And how to deal with null cells is discussed in detail in Section 4. Experiment results are described in Section 5. Finally, in Section 6, we draw conclusions and suggest future works.

2. Related work

Related literature can be grouped into four categories: The first category describes data access control. The second category is about cardinality-based security control in data cubes. In addition, the third category addresses privacy preserving data mining. Finally, a fourth category introduces a summary of security-control in statistical databases.

2.1. Data access control

Data access control is a traditional technique of protecting data in the data warehouse. Data access control is based on the notion of privileges – the authorization to perform a particular operation. Privileges are required to gain access to information in the data warehouse. However, it is well known that data access control alone is insufficient in controlling information leakage, because information not released directly may be inferred indirectly by manipulating legitimate queries about aggregated information.

Moreover, in data warehouses, data access control is not defined in terms of tables, but dimensions, hierarchical paths and granularity levels. Hence, data access control may bring up inconsistent problems [5].

2.2. Cardinality-based security control

Wang et al. [6] proposed a cardinality-based security control in data cubes. By exploring special structures of data cube operator, such as group-by, cross-tab and sub-total, the author derived cardinality-based sufficient conditions for securing data in data cubes. The main idea of this method is to enforce query restriction. Only those queries, which is regarded as safe, will be answered.

Such a query restriction method can protect the data cube from information leakage through repeating queries. However, due to the query restriction, some legal queries may not satisfy the specified security requirements. As a result, these queries have to be denied or modified for resubmission. Furthermore, the overhead of judging whether a query is safe or not is unavoidable. As we know, the response time is critical in OLAP systems. Any extra procedures can potentially slow down the response time.

2.3. Privacy preserving data mining

Privacy preserving data mining [7] has drawn considerable attentions [8–11]. The basic idea of privacy preserving data mining is to preserve data privacy by adding random noise, while making sure that the random noise still preserves the 'signal' from the data so that original data distributions can still be accurately estimated. However, in data cubes, our concern is to preserve the aggregate values for range queries but not data distributions.

Recently, Kargupta et al. [12] proposed a random matrix-based spectral filtering technique to challenge the privacy preserving approaches based on random data perturbation. However, this filtering technique cannot be effectively applied to compromise our approach for the following three reasons. First, the effectiveness of this filtering technique is based on the assumption which requires that the eigenvectors of the covariance matrix of the perturbed data are orthogonal to the eigenvectors of the covariance matrix of the original data. Consider that the cubic-wise balance method is based on purposive perturbation instead of random data perturbation. Hence, this assumption can be invalid in our method. Second, if there is no prior knowledge of perturbed data distributions, which is true in our approach, this filtering approach needs to estimate the variance of the perturbed data. Even if this variance can be correctly estimated, the process for estimating the variance adds complexity and increases computation cost significantly. Finally, the filtering approach requires the computation of eigenvalues of covariance matrix. For a large amount of data, such as a data cube, this computation cost can be intractable.

2.4. Security control in statistical databases

In general, statistical databases also deal with multi-dimensional datasets and concern about statistical summarizations over the dimensions of the data sets. In the literature of statistical databases, data privacy has been extensively studied. An excellent survey is provided in [13].

The privacy preserving techniques proposed by the statistical database community can be classified into the following three general approaches: query restriction, data perturbation, and output perturbation.

(1) *Query restriction:* The query restriction category includes two techniques. The first one restricts the size of query results [14]. The second technique controls the overlap of successive queries [15].
(2) *Data perturbation:* The data perturbation category includes preserving probability distribution, such as data swapping [16] and replacing the original database with a sample from the same distribution [17]; the fixed-data perturbation, such as adding noise to numerical attributes [18] and categorical attributes [19].
(3) *Output perturbation:* The output perturbation category includes random-sample queries [20] and varying-output perturbation [21].

Query restriction approach tries to protect confidential information by enforcing certain limitations on unsafe queries. In contrast, output perturbation approach perturbs the answer to user queries. Algorithms of both approaches are enforced only after queries arrive, which will slow down the response time of the OLAP system.

The preserving probability distribution method in data perturbation category treats the statistical database as a sample from a given population that has a given probability distribution. However, in data cubes, we aim at preserving the aggregation values of range queries instead of probability distributions.

The fixed-data perturbation for numerical attributes, also known as value perturbation, is more attractive to us, because all queries are based on the perturbed data set and no overhead of query processing is introduced. Such a feature fulfills our goal of accessibility and privacy preservation.

3. Data value perturbation based privacy preserving methods

In this section, we first briefly introduce traditional random data value perturbation based privacy preserving methods and some related issues. Next, we present our cubic-wise balance data perturbation method and show why this

method provides protection for individual data cells in data cubes and can still achieve desirable results for range queries even if data perturbation is enforced. Finally, we discuss the security and other issues related to our method.

3.1. Random data value perturbation

Privacy preservation based on random data perturbation was developed by Traub et al. [18]. This method works as follows. Assume that the true value of a given attribute (e.g., sales) of an entity k is y_k. Using this method, the result of a range-sum query on y_k will be $T = \sum_{k=1}^{n} x_k$, where $x_k = y_k + e_k$, e_k is a random variable with the expectation $E(e_k) = 0$, the variance $\mathrm{Var}(e_k) = \sigma_e^2$, and $\{e_k\}$ are independent.

3.1.1. Scaling problem

The random value perturbation method replaces the actual value x_i by $x_i + \alpha$. The security level achieved by this method depends on the choice of α. If α is a random value generated in an absolute range, the random value perturbation method may encounter the *scaling problem*. For example, if the stock a customer bought is 15,000, adding 8000 to the value would be sufficient to hide the actual value. However, if perturbing 150,000 by 8000, the actual data protection would be considered limited.

An alternative approach is to use *relative value perturbation* instead of absolute value perturbation. Relative value perturbation means that the perturbation is generated from a relative value, say 20%. For instance, if the value of a cell in a data cube is x, the perturbation generated for this cell is within the range $[-|x|20\%, |x|20\%]$.

3.1.2. Query size

By the law of large numbers, the random value perturbation method can increase the range query accuracy by applying queries to large query sets. In other words, the error bound of query results depends on the size of the query sets. Hence, the query performance would be degraded dramatically when the query size becomes small.

3.2. A cubic-wise balance method

Here, we propose a cubic-wise balance method to provide privacy preserving range queries on data cubes. This method is based on data perturbation. However, unlike random data perturbation approaches, the cubic-wise balance method provides a purposive perturbation on individual cells in a data cube. Before describing our method, we first define some terminologies as follows.

3.2.1. Terminologies

Definition 1. A *data cube* Ω with d dimension is a d-dimensional array. In this array, the size of dimension i is n_i which represents the number of distinct values for this dimension. Thus, this data cube consists of $n_1 \times n_2 \times \cdots \times n_d$ cells, and each cell can be represented as $\Omega[X_1, X_2, \ldots, X_d]$, where $0 \leqslant X_i < n_i$ and $1 \leqslant i \leqslant d$.

Definition 2. The input to a *range-sum query* can be expressed as $(l_1{:}h_1, \ldots, I_d{:}h_d)$, where l_i is the lower bound of the range query and h_i is the upper bound of the query in ith dimension of the data cube, for $1 \leqslant i \leqslant d$.

A data cube is usually not fully occupied. Some cells are empty, i.e., these cells contain NULL values. Hence, a cube *sparsity* can be calculated as $\frac{\text{the number of empty cells}}{\text{the total number of cells}}$. In the real-world data warehouses, a data cube sparsity usually ranges from 60% to 90%.

Definition 3. In a d-dimensional data cube, a *unit cube* is defined as a d-dimensional sub-cube and the size of each dimension is two. In other words, there are two distinct values in each dimension.

Definition 4. A cell $\Omega[x_1, x_2, \ldots, x_d]$ in a unit cube is the *anchor cell* if all others cells $\Omega[y_1, y_2, \ldots, y_d]$ in this unit cube satisfy $x_i \leqslant y_i \leqslant x_i + 1$, where $1 \leqslant i \leqslant d$.

Example 1. Fig. 1 gives an illustration of the concepts of data cube, query range, and unit cube. We observe that

- this is a 2-dimensional data cube and the size of each dimension is 6;
- cells: $\Omega[3,1] = 102$, $\Omega[5,3] = 41$;
- the cube sparsity $= \frac{22}{6\times 6} = 61\%$;
- the rectangle with thicker lines indicates the query result of the range query $Q(1{:}4, 1{:}5)$;
- the three shaded areas are unit cubes. For example, $\{\Omega[3,3], \Omega[3,4], \Omega[4,3], \Omega[4,4]\}$ is a unit cube and $\Omega[3,3]$ is the anchor cell of this unit cube.

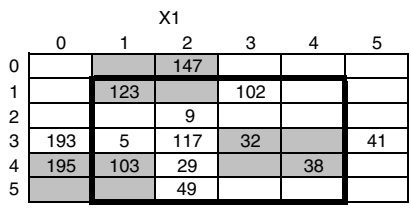


Fig. 1. An illustration of the concepts of data cube, query range, and unit cube.

3.2.2. 2-Dimension data cubes

The key idea of the cubic-wise balance method is illustrated as follows. In a unit cube, for each perturbation value, we have a counter-value to cancel its effect, so that the summation of the perturbation within the unit cube is maintained at zero. In this section, we present how the cubic-wise balance method works on 2-dimensional data cubes.

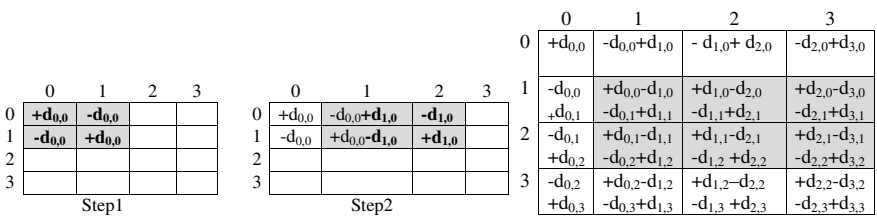
The working mechanism of the cubic-wise balance method for perturbing 2-dimensional data with size $m \times n$ is described as follows:

- Suppose $\Omega[x_1, x_2]$ is a non-null cell of a 2-dimensional data cube, where $0 \leqslant x_1 < m$, $0 \leqslant x_2 < n$, and $\Omega[x_1, x_2] = t$. Let $\Omega[x_1, x_2]$ be an anchor cell which decides a unit cube, then other cells in the same unit cube are $\Omega[y_1, y_2]$, where $x_1 \leqslant y_1 \leqslant x_1 + 1$, and $x_2 \leqslant y_2 \leqslant x_2 + 1$. Let the relative perturbation range be $[0, \Delta]$, where $\Delta > 0$. For each unit cube, this method generates a random data α where $-|t|\Delta \leqslant \alpha \leqslant |t|\Delta$ and assigns $+\alpha$ to the anchor cell $\Omega[x_1, x_2]$. The method assigns $+\alpha$ to $\Omega[y_1, y_2]$ if $((y_1 + y_2) - (x_1 + x_2)) \bmod 2 = 0$, otherwise assigns $-\alpha$ to $\Omega[y_1, y_2]$.

Example 2. Fig. 2 shows a 4×4 2-dimensional data cube. Assume that all cells are non-null cells and the relative perturbation range is $[0, \Delta]$ where $\Delta > 0$.

Let us start from cell $\Omega[0,0]$, which decides a unit cube containing four cells: $\Omega[0,0]$, $\Omega[0,1]$, $\Omega[1,0]$, and $\Omega[1,1]$. If $d_{0,0}$ is a random value generated within the range $[-|\Omega[0,0]|\Delta, |\Omega[0,0]|\Delta]$, the perturbation added to this unit cube is shown on the left side of Fig. 2(a). In the figure, the gray area indicates the unit cube. Next, we consider cell $\Omega[1,0]$, which decides a unit cube shown as the gray area on the right side of Fig. 2(a), where $d_{1,0}$ is a random value generated within the range $[-|\Omega[1,0]|\Omega, |\Omega[1,0]|\Delta]$.

Once we have the same perturbation process for all the cells, the perturbed data set is shown in Fig. 2(b). As illustrated, unlike the random value perturbation, the perturbation approach in the cubic-wise balance method is not



(a) The First Two Steps. **(b)** Perturbation Results.

Fig. 2. An illustration of the cubic-wise balance method on 2-dimensional data cube.

random but a constraint perturbation. As shown in Fig. 2(b), after perturbation, each cell is added several perturbations and the final perturbation of a cell is the summary of all these perturbations. Therefore, different cells end up with very different perturbation values.

Now the question is how the cubic-wise balance method guarantees high accuracies for range-sum queries? To illustrate this, let us consider a range-sum query $Q(1{:}3, 1{:}2)$ on the perturbed data shown in Fig. 2(b). In the figure, the query range is indicated by the gray area. Assume that $S(Q)$ is the true answer for the query Q based on actual data and $S'(Q)$ is the answer based on perturbed data, then $S'(Q) - S(Q) = d_{0,0} - d_{3,0} - d_{0,2} + d_{3,2}$, since most of the perturbations on the cells belonging to this query are cancelled with each other. As a result, the query result on the perturbed data is very close to the query result on the actual data, especially for large queries.

3.2.3. Multi-dimensional data cubes

The cubic-wise balance method can be easily extended to deal with d-dimensional data. In this case, the unit cube is also d-dimensional and the size of each dimension is two. In order to cancel the effect of perturbation added to one cell, the perturbation values assigned to any two neighboring cells have opposite signs. More specifically, if the perturbation $+\alpha$ is assigned to $\Omega[x_1, x_2, \ldots, x_d]$ which is the anchor cell, then the perturbation $+\alpha$ is assigned to $\Omega[y_1, y_2, \ldots, y_d]$ if $(\sum y_i - \sum x_i)$ is an even number; otherwise, $-\alpha$ is assigned to $\Omega[y_1, y_2, \ldots, y_d]$.

Fig. 3 shows the pseudocode of the perturbation algorithm of the cubic-wise balance method. Essentially, this algorithm is an iterative process. For each iteration, all cells in an unit cube are perturbed. Non-anchor cells are handled separately as shown in Fig. 4.

3.2.4. Error bound analysis

In this subsection, we present the error bound analysis of the cubic-wise balance method. Error bound refers to the maximum possible difference between the actual result of a range-sum query Q and the result of the same range-sum query on the perturbed data cube.

Theorem 1. *Given a 2-dimensional dataset, suppose all perturbations $d_{i,j}$ are random data uniformly distributed within $[-\alpha, \alpha]$ where $\alpha > 0$. Given a range query $Q(l_1{:}h_1, l_2{:}h_2)$, $S(Q)$ denotes the summation of query Q before perturbation while $S'(Q)$ denotes the summation of the same query, then $|S(Q) - S'(Q)| \leqslant 4\alpha$.*

Proof. As illustrated in Fig. 2(b), we have known that the final perturbation $D_{i,j}$ of cell $[i,j]$ is the summation of several perturbations:

$$D_{i,j} = d_{i-1,j-1} - d_{i-1,j} - d_{i,j-1} + d_{i,j}$$

Input: A d-dimensional data cube and a relative perturbation range Δ.

Output: A perturbed data cube.

Parameter: suffix[0..d-1]:*the current location of a cell*; curr_dim:*the current dimension*; dim_size[0..d-1]:*the size of each dimension*

Procedure Perturb(suffix, curr_dim)

```
1.  If (curr_dim==d-1) {
2.      ω ← the value of the anchor cell identified from suffix;
3.      If (ω is null) return;
4.      α ← get_perturbation(ω, Δ); //generate perturbation data value.
5.      Non_anchor_cells(suffix, suffix1, α); //deal with non-anchor cells. Details refer to Figure 4
6.      return; }//End If
7.  For ( i = 0; i<dim_size[curr_dim]-1; i++) {
8.      suffix[curr_dim] ← i;
9.      Perturb(suffix, curr_dim+1); }//End for
```

Fig. 3. The perturbation algorithm of the cubic-wise balance method.

Then

$$
\begin{aligned}
&S'(Q) - S(Q) \\
&\quad = \sum_{i=l_1}^{h_1}\sum_{j=l_2}^{h_2} D_{i,j} = \sum_{i=l_1}^{h_1}\sum_{j=l_2}^{h_2}(d_{i-1,j-1} - d_{i,j-1} - d_{i-1,j} + d_{i,j}) \\
&\quad = \sum_{i=l_1}^{h_1}\{(d_{i-1,l_2-1} - d_{i,l_2-1} - d_{i-1,l_2} + d_{i,l_2}) + (d_{i-1,l_2} - d_{i,l_2} \\
&\qquad - d_{i-1,l_2+1} + d_{i,l_2+1}) + \cdots + (d_{i-1,h_2-1} - d_{i,h_2-1} - d_{i-1,h_2} + d_{i,h_2})\} \\
&\quad = \sum_{i=l_1}^{h_1}(d_{i-1,l_2-1} - d_{i,l_2-1} - d_{i-1,h_2} + d_{i,h_2}) \\
&\quad = (d_{l_1-1,l_2-1} - d_{l_1,l_2-1} - d_{l_1-1,h_2} + d_{l_1,h_2}) \\
&\qquad + (d_{l_1,l_2-1} - d_{l_1+1,l_2-1} - d_{l_1,h_2} + d_{l_1+1,h_2}) + \cdots \\
&\qquad + (d_{h_1-1,l_2-1} - d_{h_1,l_2-1} - d_{h_1-1,h_2} + d_{h_1,h_2}) \\
&\quad = d_{l_1-1,l_2-1} - d_{l_1-1,h_2} - d_{h_1,l_2-1} + d_{h_1,h_2}
\end{aligned}
$$

Since $-\alpha \leqslant d_{i,j} \leqslant \alpha$, we get

$$|d_{l_1-1,l_2-1} - d_{l_1-1,h_2} - d_{h_1,l_{2-1}} + d_{h_1,h_2}| \leqslant 4\alpha$$

Thus, $|S(Q) - S'(Q)| \leqslant 4\alpha$. $\square$

```
Input:      An unit cube
Output:     A perturbed unit cube
Parameter: anchor_suffix[0..d-1]:the suffix of the anchor cell; curr_dim:the current dimension;
           suffix1[0..d-1]:the suffix of non-anchor cell; perturbation:return value of line 4 in Figure 3
Procedure: Non_anchor_cells(anchor_suffix,suffix1,curr_dim, perturbation)
1.  If(curr_dim==d-1) {
2.      For(i =0; i < d-1; i++) { sum1 += anchor_suffix[i]; sum2 += suffix1[i];}
3.      If(mod(sum2-sum1)!=0) perturbation = -perturbation;
4.      add perturbation to corresponding cell;
5.      return; }//End If
6.  For(j = anchor_suffix[curr_dim]; j≤anchor_suffix[curr_dim]+1; j++) {
7.      suffix1[curr_dim] = j;   curr_dim++;
8.      Non_anchor_cells(anchor_suffix,suffix1,curr_dim); }//End For
```

Fig. 4. The algorithm for data perturbation in non-anchor cells of an unit cube.

Theorem 1 indicates that, for any range-sum query over 2-dimensional data cubes, the error bound is a constant value, which is solely determined by the upper bound of the perturbation range.

For 3-dimensional data cubes, we can also derive the final perturbation value of cell $[i,j,k]$ as

$$D_{i,j,k} = -d_{i-1,j-1,k-1} + d_{i-1,j-1,k} + d_{i-1,j,k-1} + d_{d,j-1,k-1} - d_{i,j,k-1} - d_{i,j-1,k-1} - d_{i-1,j,k} + d_{i,j,k}$$

Similarly, we can derive $|S(Q) - S'(Q)| \leqslant 2^3\alpha$. In addition, for a perturbed d-dimensional dataset, the error bound of a range-sum query is given by the following corollary.

Corollary 1. *For a d-dimensional dataset, under the assumption of Theorem* 1, $|S(Q) - S'(Q)| \leqslant 2^d\alpha$.

3.2.5. Query size

As previously noted, for the random data perturbation method, the range-sum query accuracy depends on the query size. A larger query size usually leads to more accurate query results. In contrast, the cubic-wise balance method is not only able to protect the actual value by keeping a large perturbation in individual cells, but also guarantee high accuracy for range-sum queries regardless

of whether the query size is large or small. The reason is that the effect of perturbations will be cancelled out for all unit cubes contained in a range query result.

3.2.6. Security control

The cubic-wise balance method is a data value perturbation based method. Data access is through the perturbed data. Hence, the snoopers cannot improve their estimation of the value of an individual cell by repeating queries [13].

Also, there are several techniques which can estimate the actual values from partial information:

- It was shown in [22] that it is possible to fully recover original distribution from non-overlapping, contiguous partial sums, if the distribution is 'smooth'.
- There is work on estimating attribute distributions from partial information [23], or on approximating queries on sub-cubes by higher-level aggregations [24].
- Another potential security attack can come from the known marginal information. For example, in the 2-dimensional case, the correct values of sums of rows and the sums of columns are known.

However, the above techniques do not deal with information that has been deliberately perturbed. In our method, because the data access is based on the perturbed data cube, the individual cell values being estimated by partial information are perturbed values but not actual values. Hence, the above techniques cannot effectively compromise our cubic-wise balance method.

4. Handling null cells

In reality, the sparsity of data cubes is usually between 60% and 90%. Therefore, more than half of data cells should contain null values. If the cubic-wise balance method is applied to perturb a data cube without considering the presence of null cells, the sparsity of the perturbed data cube will decrease to 0. This will lead to high storage cost. In order to keep the sparsity of original data cubes, null cells must be taken into the consideration.

Example 3. Given a 2-dimensional unit cube, the anchor cell is at the up-left corner. In the unit cube, there are two null cells and two non-null cells, as shown in Fig. 5(a). Suppose the perturbation value for this unit cube is α.

Fig. 5(b) is the perturbation result without considering the presence of null cells. As shown, two null cells become non-null cells with value $+\alpha$.

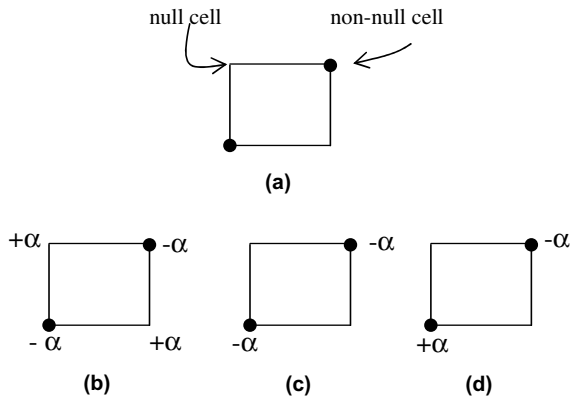


Fig. 5. 2-Dimensional unit cube with two null cells.

One simple method to deal with null cells is to let null cells still be null after the perturbation process. The result is shown in Fig. 5(c). The perturbations added to the unit cube cannot cancel each other. To deal with this problem, we change one $-\alpha$ to $+\alpha$. The result is shown in Fig. 5(d), the sum of perturbations added to this unit cube is still zero.

Since the perturbation values added to a specific unit cube are the same, only the signs (+ or −) of values are different. The problem of how to efficiently counterbalance the perturbations added to a unit cube is actually equivalent to the problem of how to assign the sign of perturbations.

4.1. Sign assignment problem

The problem on *how to optimally assign the sign of perturbations to a unit cube* can be described as follows.

Sign Assignment Problem (SAP): Given a subset S of a d-dimensional unit cube, the SAP is to find an assignment of signs to S, such that the absolute value of the sum of signs for all range queries is minimal.

Let Q be a range query in a unit cube. The absolute value of sum of signs of all cells in the query Q is equal to $H_s(Q)$:

$$H_s(Q) = \left| \sum_{\omega_i \in Q} \text{sign}(\omega_i) \right| \tag{1}$$

where $\text{sign}(\omega_i) = +1$ if ω_i is assigned a positive sign, otherwise $\text{sign}(\omega_i) = -1$. Therefore, the problem can be formulated as the following optimization:

$$\min_{\{\text{sign}(\omega_i)|\omega_i \in S\}} \left(\sum_{\text{all}Q} H_s(Q) \right)$$

If an assignment for S satisfies $H_s(Q) = 1$ if the size of Q is odd and $H_s(Q) = 0$ if the size of Q is even, this assignment is a strict optimal problem. However, the strict optimal solution may not always be achievable.

Example 4. Let $S = \{(1,1,0), (1,0,1), (0,1,1), (1,1,1)\}$. Fig. 6(a) is the result of simply forcing the original null cells to be null after the perturbation process. Fig. 6(b) shows a better solution to S. In Fig. 6(a), there are three + and only one −, thus the sum of perturbations in the unit cube is $+2\alpha$ (suppose α is the perturbation value). However, in Fig. 6(b), the number of + and − is the same, so the sum of perturbations in the unit cube is zero.

Although Fig. 6(b) is an optimal solution, it is not a strict optimal solution, since there exists a range query $Q(1,1,0{:}1)$ such that $H_s(Q) = 2$.

4.2. NF-completeness of the SAP

Here, we illustrate the NP-completeness of the SAP. By suitable transformation, SAP is equivalent to Partial Parallel Searchability (PPS) problem which was proposed by Srinivasan [25].

PPS problem: Suppose a database consists of S_t segment types. Given a set of queries Q, each of which requires different segment types of data from S_t, find an optimal distribution of S_t into k nodes, such that the number of searches to answer all queries once is minimum.

As we have known that SAP is to find an optimal sign assignment of S such that the number of sum of signs for all queries is minimal. Thus, SAP is equivalent to the PPS problem with two nodes where the segment types S_t corresponds to the subset S of a unit cube, and the two different signs are mapped to two nodes.

It has been proved in [25] that PPS is an NP-complete problem. Since SAP is equivalent to the PPS problem, SAP is also an NP-complete problem.

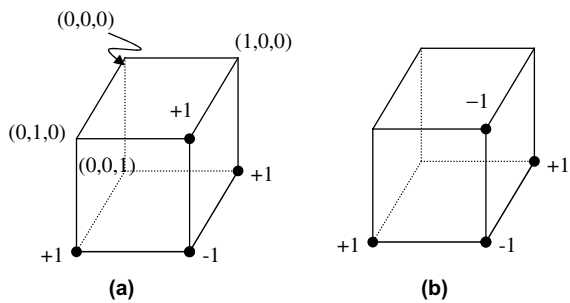


Fig. 6. 3-Dimensional unit cube with four null cells.

4.3. A heuristic assignment algorithm for SAP

Since the SAP is an NP-complete problem, we propose a Heuristic Assignment Algorithm (HAA) to deal with this problem.

The HAA is based on the observation of 2-dimensional plane:

(1) If and only if there are only two non-null cells and the two cells are at the ends of a diagonal line (as shown in Fig. 5(a)), the two cells should be assigned opposite signs.
(2) In order to minimize the sign sum of a query, two neighboring cells should be assigned opposite signs.

So the HAA first deals with all 2-dimensional planes of a range query, then process the rest unassigned non-null cells whose signs will be decided by the neighboring cells. The pseudocode of HAA is presented in Fig. 7.

The computation cost of the HAA algorithm is mainly on examining all 2-dimensional planes of a d-dimensional unit cube. Thus, the complexity of HAA algorithm is bounded by $2^{d-2}C_d^2$, which is the number of 2-dimensional planes. Since it is unnecessary to apply HAA algorithm to null unit cubes (all cells of the unit cube are null), the sparser the data cube is, the smaller the overhead of the HAA algorithm will have.

To measure the performance of the HAA algorithm, we provide a baseline algorithm, called the Basic Assignment Algorithm (BAA), which simply forces null cells to be null after the perturbation process (as shown in Fig. 5(c)). BAA is simple, but the two opposite signs assigned to non-null cells cannot be counterbalanced with each other and potentially will degrade the query accuracy.

To evaluate the performance of sign assignment algorithms, we employed a measure $\overline{H_s}(Q^d)$:

$$\overline{H_s}(Q^d) = \frac{H_s(Q^d)}{\text{number of queries}} \tag{2}$$

where

$$H_s(Q^d) = \sum_{\text{all}Q^d} \left| \sum_{\omega_i \in Q^d} \text{sign}(\omega_i) \right| \tag{3}$$

For a given unit cube, $H_s(Q^d)$ calculates the sum of all signs of all d-dimensional range queries (denoted as Q^d). $\overline{H_s}(Q^d)$ indicates the average sign sum of all d-dimensional queries. Therefore, the smaller $\overline{H_s}(Q^d)$ is, the better the solution is.

In addition, we compare $H_s(Q)$ (defined in Formula 1) of HAA and BAA for each query and see how many queries favor BAA or HAA, respectively.

Input: A unit cube Ω

Output: Sign assignment for the unit cube

Procedure: HAA(Ω)

1. For each cell ω_i, let sign(ω_i) = 0
2. **For** each 2-dimensional plane of a unit cube {
3. If (there are only two non-null cells ω_i, ω_j) and ($|\omega_i \oplus \omega_j| = 2$)
   /*the two cells lie at the ends of a diagonal line*/
4. If (the two cells haven't been assigned a sign)
5. Then {assign + and - to two cells respectively}
6. Else if (one cell has been assigned a sign)
7. Then {assign an opposite sign to the other cell} }//**End For**
8. **For** each of the rest unassigned non-null cell ω_i
9. If (the cell is the last one)
10. Then {If ($\sum sign > 0$) //($\sum sign$ *is the sum of all signs*
11. Then {sign(ω_i) = -1}
12. Else {sign(ω_i) = +1}}
13. Else {sum_s = $\sum sign(\omega_j)$ where $|\omega_i \oplus \omega_j| = 1$
14. If (sum_s > 0) Then {sign(ω_i) = -1}
15. Else {sign(ω_i) = +1}}

End For

Fig. 7. The Heuristic Assignment Algorithm (HAA) for SAP.

For a specific query, if $H_s(Q)_{\mathrm{BAA}} > H_s(Q)_{\mathrm{HAA}}$, HAA is better; if $H_s(Q)_{\mathrm{BAA}} < H_s(Q)_{\mathrm{HAA}}$, BAA is better; otherwise they perform equally. For example, Fig. 5(a) is a subset of a 2-dimensional unit cube, $S = \{(0,1), (1,0)\}$. With BAA, the sign assignment for S is shown in Fig. 5(c), and with HAA, the sign assignment for S is shown in Fig. 5(d). Given a range query $Q = (0{:}1, 0{:}1)$, the absolute value of sum of signs for BAA is 2 while the absolute value of the sum of signs for HAA is 0. Thus, for the query Q, HAA is considered better than BAA.

Case study. For a 3-dimensional unit cube, there are 2^3 cells, each cell could be null or non-null. In total, there are $2^{2^3} = 256$ possible cases. In other words, a 3-dimensional unit cube has 256 distinct subsets.

Furthermore, for a given 3-dimensional unit cube, $|Q^1| = 12$ (there are 12 1-dimensional queries), $|Q^2| = 6$ and $|Q^3| = 1$.

We calculate $H_s(Q^d)$ for all possible d-dimensional range queries over all possible 3-dimensional unit cubes.

Table 1 shows the performance of BAA and HAA for 3-dimensional unit cubes. It showed that HAA performs better than BAA except for 1-dimensional query. For all 3-dimensional queries, the average value of the sign sum of a query was only 0.5 when using HAA, but the value increased to larger than 1 when using BAA.

Table 2 shows the performance of each algorithm on queries. Although BAA was slightly better for 1-dimensional queries, HAA performed much better for higher dimensional queries. 11.72% of 2-dimensional queries favored HAA while only 4.69% of 2-dimensional queries favored BAA.

Tables 3 and 4 present the performance results of 4-dimensional unit cube. As the number of query dimension increased, the performance of HAA improved significantly. For 4-dimensional query, if using BAA, the average sign sum of a query was 1.57. This was reduces to 0.57 if using HAA. Also over 40% queries favored HAA while only 0.77% queries favored BAA.

Table 1
$\overline{H_s}$ of 3-dimensional unit cube

$\overline{H_s}$	Q^1	Q^2	Q^3
BAA	0.5	0.75	1.09
HAA	0.57	0.61	0.5

Table 2
Queries of 3-dimensional unit cube

	Q^1	Q^2	Q^3
BAA is better	3.68%	−4.69%	0
HAA is better	0	11.72%	28.91%

Table 3
$\overline{H_s}$ of 4-dimensional unit cube

$\overline{H_s}$	Q^1	Q^2	Q^3	Q^4
BAA	0.5	0.75	1.09	1.57
HAA	0.59	0.65	0.72	0.57

Table 4
Queries of 4-dimensional unit cube

	Q^1	Q^2	Q^3	Q^4
BAA is better	4.61%	6.19%	5.84%	0.77%
HAA is better	0	10.76%	24.32%	43.96%

5. Experimental evaluation

In this section, we present an experimental evaluation of the cubic-wise balance method. After a brief description of the experimental setup and performance evaluation measures, we present the performance of the cubic-wise balance method with respect to parameters including data perturbation range, data cube sparsity, and query size.

Experimental data sets. The experimental data sets were generated using the APB Benchmark program from [26]. These data sets had four dimensions: customer, product, channel, and time. The size of each dimension was 900 (customer), 9000 (product), 9 (channel), and 17 (time), respectively. The measure of interest was dollar, with range $[0, 699]$ and data type *short*. Cells with -1 dollar value were empty cells and were treated as missing or uncounted. The overall sparsity of the data cube was 20%. Hence, the file size of the data cube was $900 \times 9000 \times 9 \times 17 \times 0.2 \times 2 = 495{,}720{,}000$ bytes = 0.49 GB. The original size of the data file generated by the APB Benchmark was around 15G (ASCII text). We read each record in this file and then filled in the corresponding cells in the data cube.

Experimental platform. Our experiments were performed on a PC with a Pentium III 733 MHz, 40G hard disk, and 256 Mbytes of memory.

5.1. Evaluation measures

In this paper, we employed two measures: *privacy* and *the range query accuracy* for evaluating the performance of the cubic-wise balance method.

5.1.1. A measure of privacy

The measure of privacy should indicate how closely the actual value can be estimated. Since the final perturbation added to a cell by the cubic-wise balance method is the sum of several perturbations, the density function of the final perturbation is unknown. We can only estimate the amount of privacy at cell i by using the difference between x_i (the actual value) and y_i (the value after perturbation). Based on expectation and large number theory, we define a privacy factor to measure how close the actual value can be estimated as follows:

$$F_p = \frac{1}{N} \sum_{i=1}^{N} |y_i - x_i| \tag{4}$$

In Eq. (4), N is the total number of cells, the *privacy factor* F_p is considered as the amount of privacy. The amount of privacy calculated in Eq. (4) does not take into account the value of the actual data. To accurately quantify privacy,

we need a method which takes such information into account. A modified version of formula Eq. (4) is given below:

$$F_c = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - x_i|}{|x_i|} \tag{5}$$

The value F_c is the average amount of perturbations (relative to x-value), and is considered as the amount of *relative privacy*. Accordingly, the *relative privacy* is also modified by a factor x, at cell i, that is, a cell with actual value x and relative privacy will generate perturbations that are (uniformly) distributed over an interval of $[-|x|\Delta, |x|\Delta]$. For example, if $F_c = 50\%$, it means that in average, the difference between actual values and perturbed values is about 50% of the actual values.

5.1.2. A measure of accuracy

The difference between the sum of the perturbed values and the actual values over a range query Q is referred to as the *accuracy loss* of Q. Let *true_sum* be the sum of all actual values of cells in query Q, and *answer* be the sum of all perturbed values. Formula (6) defines the *relative accuracy loss* of Q, d_Q. As a result, we define a measure of accuracy as Eq. (7) shows:

$$d_Q = \left|\frac{answer - true\_sum}{true\_sum}\right| \tag{6}$$

$$F_{a,Q} = \frac{1}{1 + d_Q} \tag{7}$$

Note that the *accuracy factor* $F_{a,Q}$ lies between 0 and 1 (inclusive). If the accuracy factor is close to one, the range query has a high query accuracy. If the accuracy factor is close to zero, the query has a low query accuracy. When the estimated *answer* equals to the *true_sum*, accuracy factor is 1 (100%).

5.2. Experimental design

There are two major issues related to the experimental design as follows:

- *Data perturbation range*. We applied relative perturbation range to generate perturbations. For example, if the relative perturbation range is $[0, 50\%]$, and suppose a cell value is α, the perturbation of this cell is uniformly and randomly generated within the range $[-50\%|\alpha|, 50\%|\alpha|]$.
- *The baseline of performance*. We compared our method with the random value perturbation method proposed in [18]. We call such a value

perturbation as *traditional perturbation*. With the traditional perturbation method, the perturbation is relative to the value of a cell, so perturbations added to cells with different values are different.

5.3. Experimental results

In this section, we present experimental results to show the performance of the cubic-wise balance method in terms of privacy preservation and accuracy achieved.

5.3.1. The effect of data perturbation range

First, we illustrate the effect of different perturbation ranges on privacy preservation. In this experiment, we fixed the data cube sparsity at 60% and fixed the lower bound of perturbation range at zero while increasing the upper bound of perturbation range from 10% to 100%. Then we observed the achieved average privacy.

Fig. 8 shows the privacy performance of the cubic-wise balance method and the traditional perturbation method when the upper bound of perturbation range is increased. In the figure, it was not surprising to see a trend that the privacy values of both methods were increased with the increase of the upper bound of the relative perturbation ranges, since higher perturbation can bring better privacy preservation. Also, we observed that the achieved privacy of the cubic-wise balance method was systematically better than that of the traditional perturbation method. We also observed that the change of perturbation ranges affected the achieved privacy of the cubic-wise balance method much more than the traditional perturbation method. When the perturbation range increased

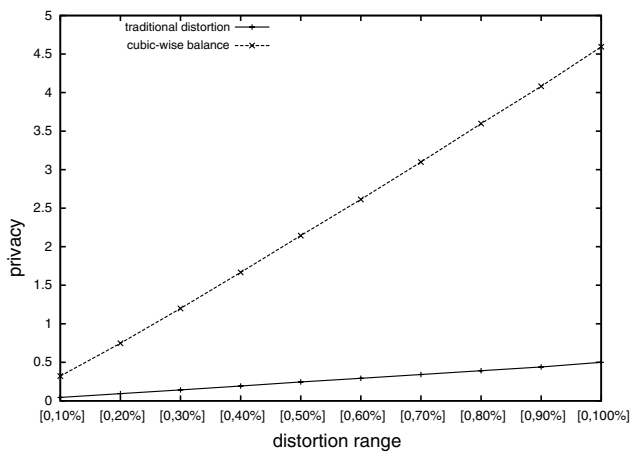


Fig. 8. The effect of perturbation ranges on privacy.

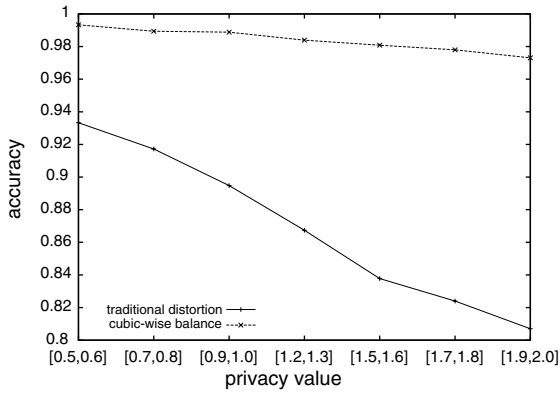


Fig. 9. The effect of privacy values on accuracy.

from $[0, 10\%]$ to $[0, 100\%]$, the privacy of the traditional perturbation method was increased from 0.04 to 0.5, while the privacy of the cubic-wise balance method was increased from 0.3 to 4.6. Finally, with the same perturbation range, the privacy of the cubic-wise balance method was much higher than that of the traditional perturbation method, especially when the perturbation range was large.

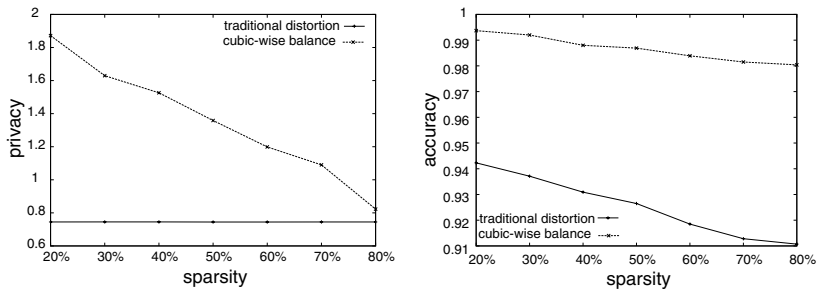
5.3.2. The interactive effect between privacy preservation and query accuracy

This experiment compared the query accuracy between the traditional perturbation method and the cubic-wise balance method at the same level of privacy. Because it was almost impossible to get exactly the same privacy value for different methods, we treated all privacy values falling into the same small range as the same level. For example, if the privacy values of different methods fall into the same small range $[0.8, 0.9]$, we considered these values were at the same level. In this experiment, the data cube sparsity was fixed at 60%. Also, we generated 600 range-sum queries with the query size between 50 and 1000. The average query accuracy of these queries was reported and compared.

Fig. 9 shows the accuracy of the cubic-wise balance method and the traditional perturbation method with the change of privacy values. As can be seen, the accuracy of cubic-wise balance method was significantly and systematically better than that of the traditional perturbation method. Also, with the increase of privacy values, the accuracy of the traditional perturbation method decreased dramatically. In contrast, the accuracy of the cubic-wise balance method was almost not affected by the change of privacy values.

5.3.3. The effect of data cube sparsity

To test the effect of data cube sparsity on the performance of the cubic-wise balance method, we fixed the perturbation range as $[0, 30\%]$ for the cubic-wise balance method and $[50\%, 100\%]$ for the traditional perturbation method. In



(a) Privacy preservation **(b)** Query accuracy

Fig. 10. The effect of different sparsity.

this experiment, we generated 600 range-sum queries with the query size between 50 and 1000.

Fig. 10(a) and (b) shows the achieved privacy values and accuracy values of the cubic-wise balance method and the traditional perturbation method with the change of data cube sparsity. As can be seen, the achieved privacy of the cubic-wise balance was consistently better than the privacy of the traditional perturbation method, while the accuracy of the cubic-wise balance method was significantly and systematically better than that of the traditional perturbation method at different data cube sparsity. Another observation was that, the less sparsity the data cube was, the better that the accuracy could be achieved for each method evaluated.

5.3.4. The effect of query size

Here, we illustrate the effect of changing query sizes. In this experiment, we fixed the data cube sparsity at 60% and kept privacy values at the same level,

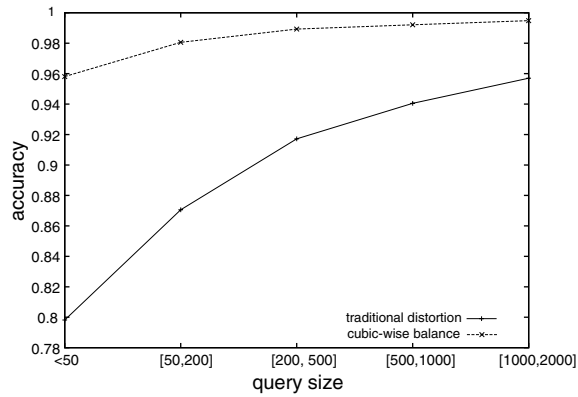


Fig. 11. The accuracy of different query size.

[0.7, 0.8]. The query size was changing from small (<50) to large (>1000). We generated 600 range-sum queries for each query size and compared the average query accuracy of these queries.

Fig. 11 shows the accuracy performance of the cubic-wise balance method and the traditional perturbation method when the query size is increased. As can be seen, the accuracy of the cubic-wise balance method was significantly and systematically better than that of the traditional perturbation method. The accuracy of the traditional perturbation method depended a lot on the query size. When the query size was small, the accuracy was as low as 80%. Only when answering large range queries, the traditional perturbation method could obtain relatively high accuracy. In contrast, the accuracy of the cubic-wise balance method was consistently over 96% even for small query size.

5.3.5. A summary of experimental results

Several major results are summarized as follows:

- The accuracy of the traditional perturbation method is very poor when the query size is small. However, the cubic-wise balance method can achieve high query accuracy even when the query size is small.
- The query accuracy of the traditional perturbation method is significantly affected by the data cube sparsity, perturbation ranges and query sizes. In contrast, the query accuracy of the cubic-wise balance method is relatively insensitive to these parameters.
- The privacy of the cubic-wise balance method is systematically better than the privacy of the traditional perturbation method. And the cube sparsity and perturbation ranges affect the privacy of the cubic-wise balance method much more than that of the traditional perturbation method.

The high accuracy achieved by the cubic-wise balance method is due to the fact that the way of cubic-wise method adding perturbations to cells enables most of the perturbations of a range-sum query be counter-balanced. In other words, the difference between the answer of a range-sum query based on perturbed data and the answer based on actual data is small, thus the accuracy loss is small.

Discussions: We now address some limitations of the cubic-wise balance method.

Since response time is important in OLAP application, we materialize data cubes for better efficiency. Indeed, materializing data cubes is not an uncommon technique to achieve better performance. Harinaryan et al. [27] stated that there are two basic approaches to facilitate OLAP. One is materializing the data cube in an multi-dimensional database (MDDB) while the raw data is in relational data warehouse. The other is to use relational data base systems and let users directly query the raw data. The MDDBs retain a significant

performance advantage, but are not very scalable, although performance in relational database systems can be improved dramatically by materializing the data cube into summary table. Therefore, we acknowledge that materializing data cubes raises the concern of scalability. The time complexity of the cubic-wise balance approach is $2^k \times n_1 \times \cdots \times n_k$ where k is the number of dimensions and n_i is the size of ith dimension of a data cube. The cost is relatively high for large data cube.

However, this is only one time expense for a data cube, and the subsequent query response time is not affected. Furthermore, once the perturbed data cube is created, the update cost is very low. Only the small block which contains the updated cell needs to be recomputed. For OLAP applications, the query response time is rather more critical. In contrast, many other privacy preserving methods, such as query restriction, require extra processing time each time when answering a query. This additional processing time will significantly slow down the response time.

In fact, the similar scalability problem also exists in pre-computing [28]. For example, the time of pre-computing may stretch in days in very large cases [29]. Parallel computing is one of the general techniques which has been used to efficiently handle the increase in data sizes [30–32]. However, the scalability of the cubic-wise balance method should be further considered in the future work.

6. Conclusions and future work

In this paper, we present a purposive value perturbation approach, called the cubic-wise balance method, for privacy preservation in data cubes. The goal is to provide an accurate estimation of the answer for range-sum queries while being able to preserve the confidential information of individual data cell in a data cube. With this method, the whole data cube is perturbed only once and all data access through the perturbed data cube rather than through the original data cube. The first benefit of this method is that no additional data access restriction is required on the perturbed data cube. Furthermore, our experimental results show that the cubic-wise balance method can achieve both better privacy preservation and better range query accuracy than the traditional random data perturbation method. Finally, as demonstrated by our experimental results, our cubic-wise balance method meets the three design goals including privacy preservation, high accuracy, and accessibility.

There are several directions for future work on this topic: Firstly, the cost of cubic-wise balance method is relatively high for higher dimensional data cube. There may be a way to improve the scalability of the proposed method. Secondly, it will be interesting to study privacy preservation in data cubes for other query types, such as range-max query and range-min query, for the sake of practical applications.

References

[1] Hackers, Crackers, Spooks, ensuring that your data warehouse is secure, DBMS Magazine 10 (4) (1997) 14.
[2] D. Kim, E. Lee, M. Kim, Y. Lee, An efficient processing of range-min/max queries over data cube source, Information Sciences 112 (1998) 223–237.
[3] P.M. Deshpande, K. Ramasamy, A. Hukla, J.F. Naughton, Caching multidimensional queries using chunks, in: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, 1998, pp. 259–270.
[4] V. Harinarayan, A. Rajaraman, J.D. Ullman, Implementing data cubes efficiently, in: Proceedings of the 15th ACM SIGMOD International Conference on Management of Data, 1996.
[5] T. Priebe, G. Pernul, Towards OLAP security design—survey and research issues, in: Proceedings of the third ACM International Workshop on Data Warehousing and OLAP, 2000.
[6] L. Wang, D. Wijesekera, S. Jajodia, Cardinality-based inference control in sum-only data cubes, in: Proceedings of the Seventh European Symposium on Research in Computer Security, 2002, pp. 55–71.
[7] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the 19th ACM SIGMOD Conference on Management of Data, 2000, pp. 439–450.
[8] A. Evfimievski, J. Gehrke, R. Srikant, Limiting privacy breaches in privacy preserving data mining, in: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2003, pp. 211–222.
[9] R.A. Evfimievski, R. Srikant, J. Gehrke, Privacy preserving mining of associations rules, in: Proceedings of the Eighth Conference on Knowledge Discovery and Data Mining, 2002, pp. 217–228.
[10] I. Dinur, K. Nissim, Revealing information while preserving privacy, in: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2003, pp. 202–210.
[11] J. Vaidya, C. Clifton, Privacy preserving association rule mining in vertically partitioned data, in: Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining, 2002, pp. 639–644.
[12] H. Kargupta, S. Datta, Q. Wang, K. Sivakumar, On the privacy preserving properties of random data perturbation techniques, in: Proceedings of IEEE International Conference on Data Mining, 2003, pp. 19–22.
[13] N.R. Adam, J.C. Wortman, Security-control methods for statistical databases: a comparative study, ACM Computing Surveys 21 (4) (1989) 515–556.
[14] D.E. Denning, P.J. Denning, M.D. Schwartz, The tracker: a threat to statistical database security, ACM Transactions on Database Systems 4 (1) (1979) 76–96.
[15] D. Dobkin, A.K. Jones, R.J. Lipton, Secure database: protection against user influence, ACM Transactions on Database Systems 4 (1) (1979) 97–106.
[16] S.P. Reuss, Practical data swapping: the first steps, ACM Transactions on Database Systems 9 (1) (1984) 20–37.
[17] C.K. Liew, U.J. Choi, C.J. Liew, A data distortion by probability distribution, ACM Transactions on Database Systems 10 (3) (1985) 395–411.
[18] J.F. Traub, Y. Yemini, H. Woznlakowski, The statistical security of a statistical database, ACM Transactions on Database System 9 (4) (1984) 672–679.
[19] A.-L. Abul-ela, B.G. Greenberg, D.G. Horvitz, A multi-proportions randomized response model, Journal of American Statistical Association 62 (319) (1967) 990–1008.
[20] D.E. Denning, Secure statistical databases with random sample queries, ACM Transactions on Database Systems 5 (3) (1980) 291–315.

[21] L.L. Beck, A security mechanism for statistical databases, ACM Transactions on Database Systems 5 (3) (1980) 316–338.
[22] C. Faloutsos, H. Jagadish, N. Sidiropoulos, Recovering data from summary information, in: Proceedings of the 23rd International Conference on Very Large Data Bases, 1997, pp. 36–45.
[23] D. Barbara, W. DuMouched, C. Faloutsos, P. Haas, J.M. Hellerstein, Y. Ioannidis, The New Jersey data reduction report, Data Engineering Bulletin 20 (1997) 3–45.
[24] D. Barbara, M. Sullivan, Quasi-cubes: exploiting approximations in multidimensional databases, ACM SIGMOD Record 26 (3) (1997) 12–17.
[25] B. Srinivasan, Parallel searching in distributed databases, Computer Networks 4 (1980) 157–166.
[26] O. Council, OLAP council's release II of the analytical processing benchmark (apb-1) for OLAP server performance, 1998. Available from: <http://www.olapcouncil.org/news/APBlr2b_PR.htm>.
[27] H. Harinarayan, A. Rajaraman, J.D. Ullman, Implementing data cubes efficiently, in: Proceedings ACM SIGMOD 1996, 1996, pp. 205–216.
[28] Y. Chen, F. Dehne, T. Eavis, A. Rau-Chaplin, Building large Olap data cubes in parallel, in: Proceedings of the International Database Engineering and Applications Symposium, 2004, pp. 367–377.
[29] Microsoft, EMC, Unisys, T3 project technical overview, White Paper.
[30] F. Dehne, T. Eavis, S. Hambrusch, A. Rau-chaplin, Parallelizing the data cube, Distributed and Parallel Databases 11 (2) (2002) 181–201.
[31] H. Lu, X. Huang, Z. Li, Computing data cubes using massively parallel processors, in: Proceedings of Seventh Parallel Computing Workshop, 1997.
[32] S. Goil, A. Choudhary, High performance OLAP and data mining on parallel computers, Journal of Data Mining and Knowledge Discovery 1 (4) (1997) 391–417.