COP: Privacy-Preserving Multidimensional Partition in DAS Paradigm

Jieping Wang[1,2], Xiaoyong Du[1,2], Haocong Wang[1,2], Pingping Yang[1,2]
1. Key Laboratory of Data Engineering and Knowledge Engineering, MOE
2. School of Information, Renmin University of China
Beijing 100872, China
{wangjieping,duyong,hcwang,yangpp}@ruc.edu.cn

ABSTRACT
Database-as-a-Service (DAS) is an emerging database management paradigm wherein partition based index is an effective way to querying encrypted data. However, previous research either focuses on one-dimensional partition or ignores multidimensional data distribution characteristic, especially sparsity and locality. In this paper, we propose **C**luster based **O**nion **P**artition (COP), which is designed to decrease both false positive and dead space at the same time. Basically, COP is composed of two steps. First, it partition covered space level by level, which is like peeling of onion; second, at each level, a clustering algorithm based on local density is proposed to achieve local optimal secure partition. Extensive experiments on real dataset and synthetic dataset show that COP is a secure multidimensional partition with much less efficiency loss than previous top down or bottom up counterparts.

Categories and Subject Descriptors
H.2.7 [**Database Administration**]: Security, integrity, and protection;

H.3.3 [**Information Search and Retrieval**]: Clustering

Keywords
Database security, Multi-dimensional partition, Cluster, DAS.

1. INTRODUCTION
With the rapid development of Internet and wide application of SOA, a novel database management paradigm DAS [1] is emerging. Different from traditional database management, wherein data owners introduce DBMS in-house, in DAS paradigm, data owners outsource their data to the third party: database service provider (DSP). Compared with traditional database management, DAS brings additional advantages to data owners, including web wide data access, high quality database service at reduced cost, relief from heavy database management routines, etc.

Although data owners outsource their data to DSP, they do not trust DSP. The untrusted of DSP brings new challenge about database security, which can be broadly divided into two categories: data privacy and data integrity. To keep privacy, data is usually encrypted before shifted to DSP, which has serious performance problem. Data integrity refers to how to keep data correctness, completeness and freshness [2]. In this paper we only focus on data privacy. To counter influence caused by encryption, much attention have been paid to querying encrypted data directly, among which partition based index is a simple but effective one [3,4,5]. Basically, partition based index consists of two steps: first, group plaintext into partitions, each of which has a unique identifier; second, build index based on partitions. In this paper we only focus on the first step and let the second step as our future work. Intuitively, there's a tradeoff between security and efficiency in partition scheme. The security of partition lies in two aspects: indistinguishable of data within the same partition, the result set from DSP is at partition level instead of tuple level. On the other hand, efficiency of partition also lies in two aspects: false positive, which refers to tuples satisfying partition based SQL while not satisfying plaintext based SQL; dead space, which refers to empty space included by any partition. Formal definition of security and efficiency loss will be presented in section 2. In this paper, we propose a secure multidimensional partition which decreases both false positive and dead space.

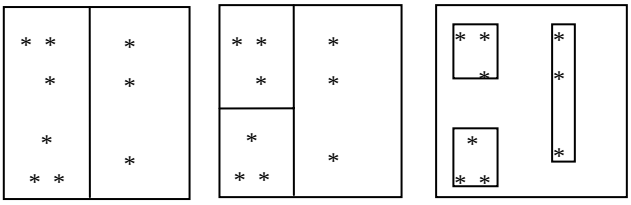

(a) One-dimensional (b) Whole space (c) Ideal

Figure 1. Comparison of three different partitions

To above problem, previous research can be grouped into two classes. The first is to build one-dimensional partition for each dimension [4], which is basically a grid partition. The second is whole space multidimensional partition scheme, such as Mondrian [9]. However, neither of them is effective to minimize efficiency loss. For example, assume the secure constraint is that each partition must have at least 3 tuples. One-dimensional based partition and whole space multidimensional partition are illustrated in Figure 1(a) and 1(b) respectively. In Figure 1(a), no further partition is available to achieve security, which results in large false positive and dead space. In Figure 1(b), efficiency loss is mainly caused by dead space. Besides, with the number of dimension increasing, the percentage of dead space increases dramatically. The ideal partition is illustrated in Figure 1(c), where each partition consists of 3 nearby points so that both false positive and dead space are minimized. To achieve ideal partition,

in this paper we propose a new multidimensional partition: **C**luster based **O**nion **P**artition (COP), which has the following advantages:

1. COP is designed to decrease both false positive and dead space;
2. COP gives special attention to multidimensional data distribution characteristics, especially sparsity and locality;
3. Compared to previous multidimensional partition, which partition space either top down or bottom up, COP is an integrated solution, which combines them together;
4. Different from previous top down partition, which divides one partition into two each time, COP forms partitions level by level, which adheres to real data distribution;
5. Based on local density an efficient clustering algorithm is proposed to achieve local optimal partition;

The rest of the paper is organized as follows. We first introduce security definition and efficiency loss measurements in Section 2. Section 3 gives related work in multidimensional partition. Section 4 describes how COP works in detail. Extensive experiments are made in section 5. Section 6 concludes the paper and gives potential directions for future work.

2. SECURITY DEFINITION AND EFFICIENCY LOSS MEASUREMENT

In order to facilitate further discussion, we introduce some notations and definitions in Table 1, which will be used throughout the paper.

Table 1. Notations and Definitions

Notation	Definition
$\boldsymbol{F^D}$	D-dimensional data space
N	# of data points in $\boldsymbol{F^D}$
M	# of cells in $\boldsymbol{F^D}$
$\boldsymbol{C^D}$	Covered space in $\boldsymbol{F^D}$
$\boldsymbol{E^D}$	Empty space in $\boldsymbol{F^D}$
$\|p\|_d$	# of data points in partition p
$\|p\|_c$	# of cells in partition p
d(cell)	Cell's value project at d_{th} dimension

In most database applications, multidimensional query takes the form of rectangle or hyper-rectangle. For brevity of illustration, we use rectangle to denote both of them. Correspondingly, we limit our multidimensional partition to rectangle. Basically, the security of rectangle lies in two factors: rectangle coverage and the number of data point within the rectangle, to which we introduce our security definition.

Definition 1: (C, K)-bound Partition. In d-dimensional space, each dimension is divided into p_i partitions ($1<=i<=d$) and the whole space is divided into $p_1*p_2*\ldots*p_d$ cells. Given minimal coverage C and minimal number of data point K, we call a partition P is (C, K)-bound if it satisfies the following conditions:

1. $|P|_c >= C$;
2. For any data point t in P, there must exist a set R with $|R|_d>=K$, so that $t \in R$, and $R \subseteq P$.

Based on above definition, a partition is called secure only when it has enough coverage and enough number of data point. In grid structure, the calculation of partition coverage can be transformed to counting the number of cells within the partition.

As we pointed out previously, efficiency loss is majorly caused by false positive and dead space, to which we introduce the following measurements:

Definition 2: Average Partition Size (APS). Assume there're N data points in $\boldsymbol{F^D}$, which are grouped into s secure partitions, denoted as P_1, P_2, …, P_s respectively. APS is calculated as follows:

$$\mathrm{APS} = \left(\sum_{i=1}^{s} \left| P_i \right|_d \right) \Big/ s \tag{1}$$

Definition 3: Normalized Partition Coverage (NPC). Assume there're N data points in $\boldsymbol{F^D}$, which are grouped into s secure partitions, denoted as P_1, P_2, …, P_s respectively. NPC is calculated as follows:

$$\mathrm{NPC} = \sum_{i=1}^{s} \left| P_i \right|_c \Big/ \left| F^D \right|_c \tag{2}$$

Here we partition F^D into O^D and E^D, which satisfy $F^D = O^D \cup E^D$. APS is used to measure efficiency loss caused by false positive. The larger APS, the larger the efficiency loss would be. In an optimal partition, APS would be minimized to K. NPC is used to measure efficiency loss caused by dead space. The larger NPC, the larger the efficiency loss would be. In an optimal partition, NPC is minimized to $\left|C^D\right|_c \Big/ \left|F^D\right|_c$ Here we use normalized partition coverage instead of average partition coverage, which is based on the following consideration: average partition coverage is heavily influenced by dimensionality while normalized partition coverage is not. Specifically, with the number of dimension increasing, the number of grid cell (M) increases dramatically, which results in both partition coverage and $|\boldsymbol{F^D}|_c$ increasing dramatically.

Intuitively, there's a tradeoff between security and efficiency. Specifically, with the increasing of K and C, security is increasing. On the other hand, efficiency is decreasing and efficiency loss is increasing, which is reflected by the increasing of APS and NPC. Based on above definitions, we introduce our multidimensional partition problem formally.

Problem Statement: Assume there're N data points in F^D, given security constraint (C, K), the problem of optimal secure multidimensional partition is to group all data points into partitions so that each partition is (C, K)-bound. At the same time, both APS and NPC are minimized as possible.

3. RELATED WORK

As far as we know, [6] proposed the first multi-dimensional partition in DAS paradigm. Basically, it's a bottom up partition, which is only efficient in 2-dimensinal space. Besides, there has been much research about multidimensional K-anonymity in data publishing and multidimensional histogram in query optimization. In k-anonymity, generalization and suppression are two commonly used techniques to keep privacy. There has been much

research work about how to achieve k-anonymity by partition, which can be categorized into three classes: full domain partition, top down multidimensional partition and bottom up multidimensional partition. Compared with the last two partitions, full domain partition, such as Incognito [8], belongs to global recoding and brings more information loss than local recoding. Interested readers could refer to [8] for more details.

The basic idea of top down approach is to divide available partition into two recursively until no further partition is possible. Among them, Mondrian [9] is a well-known one, which selects the dimension with widest range and partition based on median of selected dimension. Utility based [13] is another one, which selects two farthest points as seeds and form two partitions based on nearest neighbor. However, utility based partition is not rectangle constrained and unsuitable for our problem. The most obvious advantage with top down solution is simple and efficient. The time complexity is O(N*logK) for Mondrian and $O(N^2)$ for utility based. On the other hand, it has serious drawbacks: first, it's a whole space partition, which cause large information loss especially in sparse space; second, partition quality is heavily influenced by the selection of dimension and partition value, to which previous research has not paid much attention to.

On the contrary, bottom up partition starts with single data point and merges incrementally until security is achieved. Several greedy algorithms have been proposed including clustering based [10] and utility based [13]. In [10], a novel approach is proposed based on clustering, in which instead of publishing generalized data, cluster center is published. However, cluster in [10] is defined by circle, which is unsuitable for our problem. In utility based solution [13], for each insecure group, a linear scan of all remaining insecure groups is needed to find one with maximized utility. It has the following drawbacks: first, it's time-consuming and the time complexity is $O(n^2*\log K)$; second, incremental merging can not guarantee the final partition utility is maximized. Besides partition, there's another class of approach based on space-filling curve [17], which is beyond the scope of the paper.

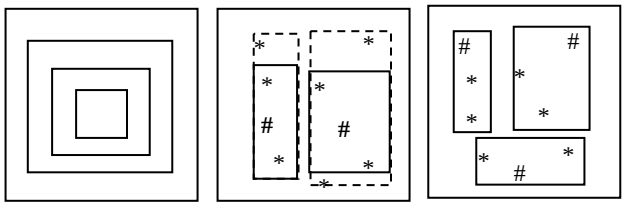
Although there has been much research in multidimensional K-anonymity, there exists some common problem: they did not consider data distribution characteristic in multidimensional space, especially sparsity and clustering, which leads to top down partition ineffective in sparse space and bottom up partition inefficient in partition merging. Besides, there's one big difference between K-anonymity with our partition: in K-anonymity, if some data (usually outlier) cause much information loss, they could be excluded by any partition; however, all the data must be included in our partition.

Another related research field is multidimensional histogram, which focuses on keeping distribution in partition as uniform as possible. There has been much research on multidimensional histogram. Interested readers can refer to [7] for an overview. According to whether partitions are overlapping or not, there're two classes: non-overlapping partition, such as equi-depth, MHist, and STGrid, overlapping partition, such as GenHist and STHoles. According to whether partition is dynamic or not, there're also two classes: dynamic histogram, such as STGrid and GenHist, static histogram, such as equi-depth and MHist. The similarity between multidimensional histogram with our partition is obvious: their partitions are all rectangle shaped. However, the distinction between them is also obvious: the focus of histogram is uniform data distribution, while the goal of our partition is to form secure partition with minimal efficiency loss. Besides, histogram is error tolerate because it's mainly used for query cost estimation, while ours is not, where any insecure partition is forbidden.

4. CLUSTER BASED ONION PARTITION

Actually, our multidimensional partition problem is a constrained optimization problem, which is NP-hard [10, 11]. From analysis in related work, it's inferred that solely top down or bottom up solution is insufficient to achieve approximately optimal partition. In this paper we propose Cluster based Onion Partition (COP) which combines them together. Specifically, COP is composed of two steps: partition level by level from outside to inside, which is a top down step; at each level, form local optimal partition based on clustering, which is a bottom up step. In the following we'll discuss them respectively.

4.1 Top Down Onion Partition



(a) Onion partition (b) Inside point first (c) Outside point first

Figure 2. Heuristic based onion partition

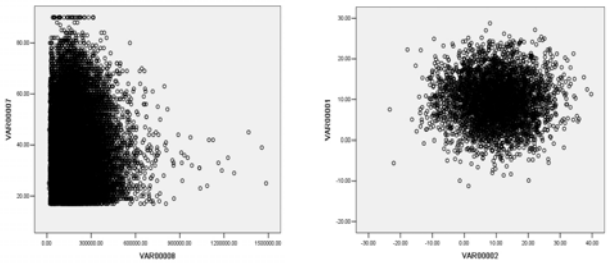


Figure 3. 2-dimensional data distribution of two data sets

Different from previous top down partition, which divides one partition into two recursively, in COP, we divide space level by level from outside to inside, which is illustrated in Figure 2(a). Since the whole partition procedure is like peeling of onion, we call it onion partition in the rest of paper. Correspondingly, previous top down partition is called half partition. By observing data distribution of various data sets, we find that space is rarely divided into two distinct parts. On the contrary, space is usually composed of clusters with various density and coverage. For example, Figure 3 plots distribution of two data sets used in experimental section. From Figure 3, it's easy to see that although data are distributed non-uniformly in the whole space, it's distributed uniformly in a local area. Specifically, they are densely distributed inside while sparsely outside, which trigger

our onion partition. The reason of outside cell first is illustrated in the following example.

Example 1. Assume security constraint (C, K) is (1, 3). The first selected points are denoted by #. In Figure 2(b), we first select inside point, which first form two real line rectangles. Then for the remaining three points, since it's expensive to form secure partition by themselves, they are merged to their nearest secure partitions respectively. The final partition is plotted as two dash line rectangles. In Figure 2(c), we first select outside point and the final partition is plotted as three real line rectangles. Obviously, both APS and NPC of partitions in Figure 2(c) are less than those in Figure 2(b).

From above example, it can be inferred that outside point first will cause less efficiency loss than otherwise. Correspondingly, we divide all cells into two classes: boundary cell (outside point) and interior cell (inside point). The formal definition of boundary cell is as follows:

Definition 4: Boundary cell. In D-dimensional space, a cell is called boundary cell if there exists at least one dimension d such that $d(cell) \in \{\min(d(cell_i), \max(d(cell_i)), cell_i \in C^D$.

To find all boundary cells, a straightforward solution is to form minimal boundary rectangle (MBR). The cells consisting of MBR form boundary cells exactly. Analysis of time complexity is as follows:

Time complexity analysis: The time of onion partition lies in two factors: the time to form MBR and the number of onion level. To get MBR, we need scan all cells linearly and the time complexity is O(M). The number of maximal onion level is $\min(p_i)/2$, where p_i is the number of partitions in i_{th} dimension. Usually, $\min(p_i)$ is a constant much less than N.

4.2 Cluster Based Partition

Algorithm: Cluster based Onion Partition (COP)

Input: D-dimensional data, security constraint (C, K)

Output: secure partition list

```
1.  Partition data into cells;
2.  Form boundary cell list;
3.  WHILE (boundary cell list is not empty)
4.      FOR each cell in boundary list;
5.          form minimal (C, K) partition;
6.          remove partition cells from boundary
        cell;
7.      END FOR;
8.      form boundary cell list of remaining cells;
9.  END WHILE;
10. output partition list;
```

Figure 4. COP Algorithm

The algorithm of COP is illustrated in Figure 4, from which we can see that after formation of boundary cells, the next step is to form minimal (C, K) partition covering all of them (line 5). Here minimal is based on the definition of APS and NPC. The naïve solution is to get all coverings of each cell incrementally until a minimal one is found. In the following, we first introduce the concept of cell covering and unit space, and then show it's a time-consuming process.

Definition 5: Cell Covering. Cell covering is a minimal bounding rectangle which consists of cell and at least one of its neighbors.

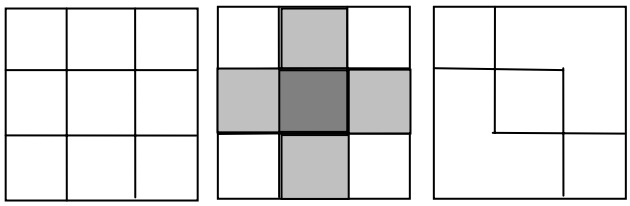
Here we assume a cell has two neighbors at each dimension. So in D-dimensional space, a cell has at most 2*D neighbors.

Definition 6: Unit space. In D-dimensional space, unit space is minimal bounding rectangle of one interior cell and its 2*D neighbors.

Theorem 1. In D-dimensional unit space, the number of covering for interior cell is 4^D-1.

Proof: In D-dimensional unit space, an interior cell has 2*D neighbors in total. A covering of interior cell is formed consisting of interior cell and its neighbor. The number of neighbor ranges from 1 to 2*D, each of which has C^i_{2*D} combination (1<=i<=2*D). So the total number of combination is $\sum_{i=1}^{2*D} C^i_{2*D} = 2^{2*D} - 1 = 4^D - 1$.

For 2-dimensional unit space, which is illustrated in Figure 5(a), interior cell and four neighbors are illustrated as dark grey and light grey cell respectively in Figure 5(b). There're C^1_4 coverings, each of which is composed of one interior cell and one neighbor. Figure 5(c) includes two coverings consisting of interior cell and two neighbors.



(a) Unit space (b) Interior cell and 4 neighbors (c) Covering

Figure 5. Covering in 2-dimensional space

Algorithm: Cluster based partition

Input: cell, security constraint K

Output: minimal secure partition

```
1.  Calculate local density of cell;
2.  Form cluster based on K and local density;
3.  WHILE (cluster is not K guaranteed)
4.      increment cluster according to local density;
5.  END WHILE;
6.  Sort cells in cluster based on partition coverage;
7.  get first cell in cluster and form partition;
8.  WHILE (partition is not secure)
9.      get next cell in cluster and form partition;
10. ENDWHILE;
11. Output partition;
```

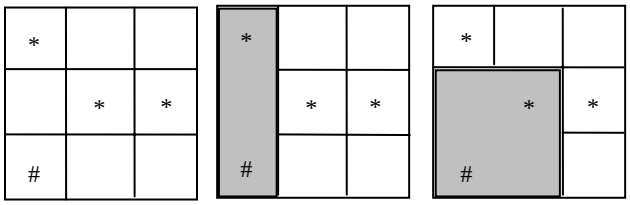
Figure 6. Cluster based minimal partition formation

To form minimal secure partition efficiently, we propose cluster based algorithm, which is illustrated in Figure 6. Cluster is an

important problem in data mining, which has many solutions, including partition based, hierarchical based and density based. In partition based solution, the number of partition must be given beforehand, which is difficult in our problem. In hierarchical based solution, data are aggregated or divided progressively, similar to bottom up or top down partition. So they have common problems. In density based solution, density is defined by two parameters: minimal distance and minimal number of data point. When data distribution is non-uniform, it's difficult to specify these two parameters globally, to which we propose a nonparametric local density based clustering algorithm.

Based on clustering algorithm, an approximately minimal cell covering is found which covers all candidate cells to secure partition. Comparably, bottom up solution will find candidate cells in the whole space. Intuitively, cluster coverage is heavily influenced by security constraint K and local density. The larger the K, the larger the cluster coverage would be. On the contrary, the larger the local density, the less the cluster coverage would be.

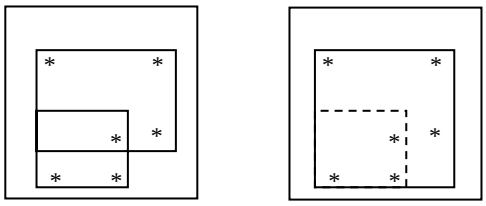
Once cluster is formed, the next step is to form minimal (C, K) partition within the cluster, to which we sort candidate cells, which has time complexity of O(k*logk). Notice here we sort candidate cells based on partition coverage instead of Euclidean distance. The reason is illustrated in Figure 7. Assume security constraint (C, K) is (1, 2). For cluster in Figure 7(a), we need to form secure partition covering cell #. According to coverage based sort, final partition is illustrated as grey rectangle in Figure 7(b). According to distance based sort, final partition is illustrated as grey rectangle in Figure 7(c). Obviously, partition in Figure 7(b) has less NPC than that in Figure 7(c).



(a) Cluster (b) Coverage based (c) Distance based

Figure 7. Comparison between two sort criteria

4.3 Discussion



(a) Overlapping partition (b) Nested partition

Figure 8. Transformation from overlapping partition to nested partition

In this subsection, we discuss another two factors related to efficiency loss. The first is partition overlapping. Obviously, there's no partition overlapping in top down partition. It only exists in bottom up partition and COP. Partition overlapping will cause large efficiency loss when query falls in overlapped area. In the following, we first introduce $\text{Ratio}_{\text{overlap}}$ to measure partition overlapping, and then propose a nested partition scheme to decrease partition overlapping.

$$\text{Ratio}_{\text{overlap}} = \sum_{i=1}^{s}\sum_{j=i}^{s}\left|P_i \bigcap P_j\right|_c \Big/ \sum_{i=1}^{s}\left|P_i\right|_c \qquad (3)$$

$\text{Ratio}_{\text{overlap}}$ ranges from 0 to 1. The less the $\text{Ratio}_{\text{overlap}}$, the less the efficiency loss would be. When $\text{Ratio}_{\text{overlap}}$ is 0, there's no overlapping. By observing the overlapping partitions in COP, we find that quite a few overlapping is like that illustrated in Figure 8(a) (assume K is 3). Traditional way of merge and split couldn't decrease overlapping in that case. So we propose a nested partition illustrated in Figure 8(b), which could decrease partition overlapping effectively. On the other hand, nested partition increases space cost.

The second factor is query workload. In cluster based partition, we assume the workload is uniform, which means the access probability of each cell is equal. However, it's not necessarily the case in most real applications. For non-uniform workload, especially skewed workload, it's preferred to ensure APS and NPC of heavily accessed area is minimized. To measure workload aware efficiency loss, we introduce false positive ratio, which is calculated as follows:

$$\text{Ratio}_{\text{FP}} = \sum_{i=1}^{w}\frac{|P(Q_i)|_c - |RS(Q_i)|_c}{|P(Q_i)|_c} \Big/ w \qquad (4)$$

In above formula, there're w queries $Q_1, Q_2, \ldots Q_w$ in total. $RS(Q_i)$ denotes result set of query Q_i. $P(Q_i)$ denotes minimal partition covering final result set. Ratio_{FP} ranges from 0 to 1. In an optimal partition, Ratio_{FP} is 0. With the increasing of Ratio_{FP}, efficiency loss increases correspondingly.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

The experiments were carried on a Pentium IV 3.2G Hz machine, with 1.5G RAM. COP is implemented in JAVA/JRE 1.6 development kit and Runtime Library.

The experiments were carried on two data sets with different data distribution. The first data set is Adults database from UCI [16], which is the de facto benchmark of K-anonymity. The table has 14 attributes, we use 6 continuous attributes of which comprising *age*, *final weight*, *education-num*, *capital-gain*, *capital-loss*, *hours-per-week*. The table has 32,561 records in total. The second data set was synthetically generated with 6 continuous attributes conforming to normal distribution with same mean and different standard variance. The synthetic data set has 100,000 records in total. The distribution of the first two attributes for two data sets is illustrated in Figure 3 respectively.

5.2 Result Analysis

In this section, extensive experiments have been done to show COP is a secure multidimensional partition with less APS and NPC than two previously proposed approaches: top down partition (Mondrian) and bottom up partition (utility based). To simplify problem, data space is initially partitioned into grid cells

in such a way that each cell is coverage secure by itself, which means we can focus on influence of K. Specifically, we divide all experiments into four groups to explore the impacts of the following dimensions respectively:

•The number of dimension. Considering the number of predicates in multi-dimensional query is usually not larger than 6, we range the number of dimension from 2 to 6 correspondingly.

•The setting of security constraint K, which is mainly decided by user specification and data density.

•The degree of overlap, which only exists in bottom up and COP partition.

•False positive ratio under specific workload.

In the first group, we analyze the impact of dimensionality on APS and NPC. The results are illustrated in Figure 9. We first analyze the influence of dimensionality on APS. In Figure 9(a) and 9(b), with the increasing of dimensionality, APS of COP almost keeps stable and the least of three in both data sets. However, APS of other two partitions is not only influenced by dimensionality, but also influenced by data distribution. Specifically, APS of top down increases dramatically with dimensionality and larger than that of bottom up in Adults data set, while it's contrary in synthetic data set. It can be inferred that COP could achieve minimal partition size independent of dimensionality and data distribution. Then we analyze the influence of dimensionality on NPC. In Figure 9(c) and 9(d), with the increasing of dimensionality, NPC of top down partition is keeps stable, while NPC of other two partitions decreases dramatically and COP is always the best of all. The reason is that top down is whole space partition, so its NPC is always 1. On the other hand, with dimensionality increasing, data density decreases, which leads to NPC of packed partitions decreasing. It's counter-intuitive that when dimensionality is less than 4, NPC of top down partition is even less than bottom up. Actually, it is caused by large amount of partition overlapping in bottom up solution, which will be analyzed in detail in the third group of experiments. Comparably, overlapping of COP is relatively little so that its NPC is always less than that of top down.

In the second group, we analyze the impact of K on APS and NPC. The results are illustrated in Figure 10, from which we can see that with the increasing of K, both APS and NPC are increasing correspondingly. It conforms to the tradeoff between security and efficiency. COP is always the best of all three solutions, which comes to the conclusion that combining top down and bottom up together is better than only using one of them. The reason that top down is always better than bottom up is that with the increasing of K, more cells are needed to form secure partition, which increases the percentage of partition overlapping.

In the third group, we compare overlapping ratio of three solutions. To be fair, we didn't use nested partition to decrease overlapping in COP. The results are illustrated in Figure 11(a) and 11(b). It's obvious that top down partition has zero overlapping ratio. As for bottom up partition, it decreases dramatically with the increasing of dimensionality. Specifically, when dimensionality is less than 4, overlapping ratio of bottom up is much larger than that of COP, which exactly explains the results in the above two groups. On the other hand, dimensionality has much less influence on overlapping ratio of COP.

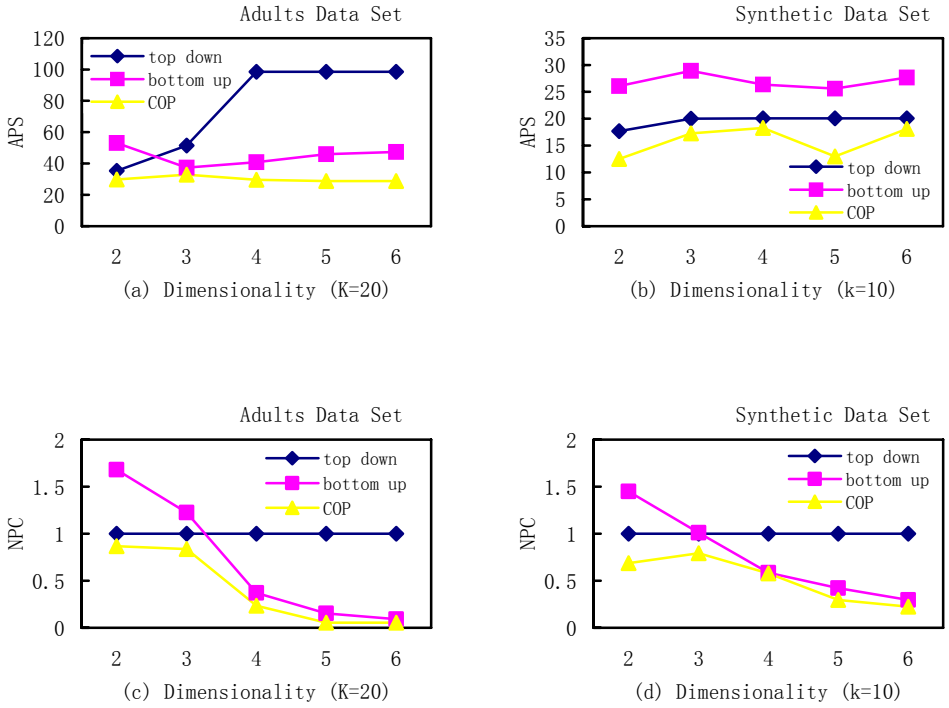


Figure 9. Impacts of dimensionality on APS and NPC

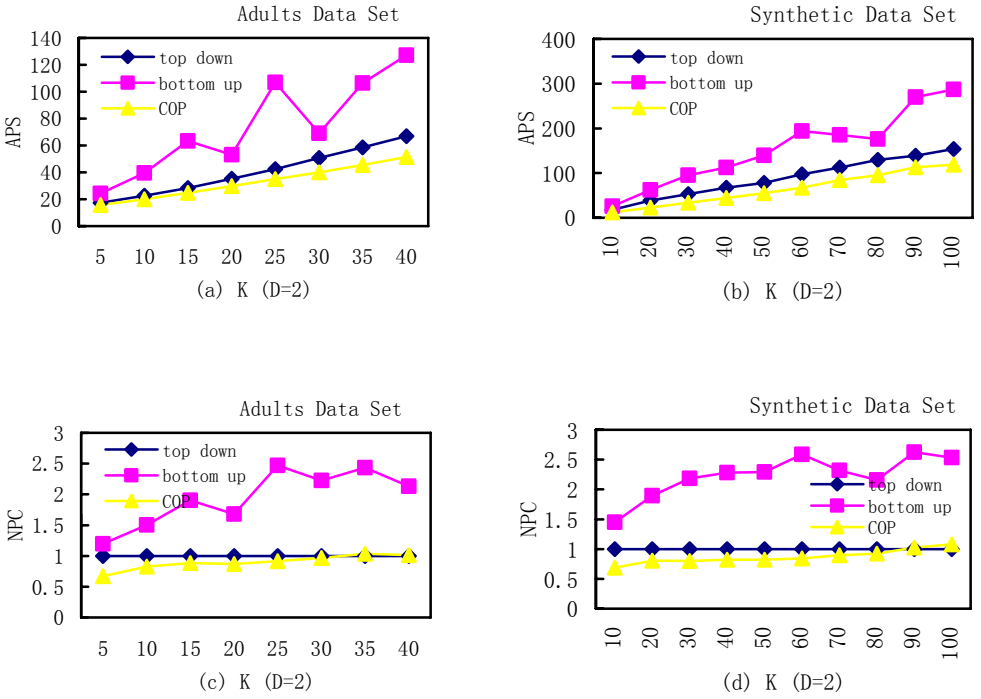
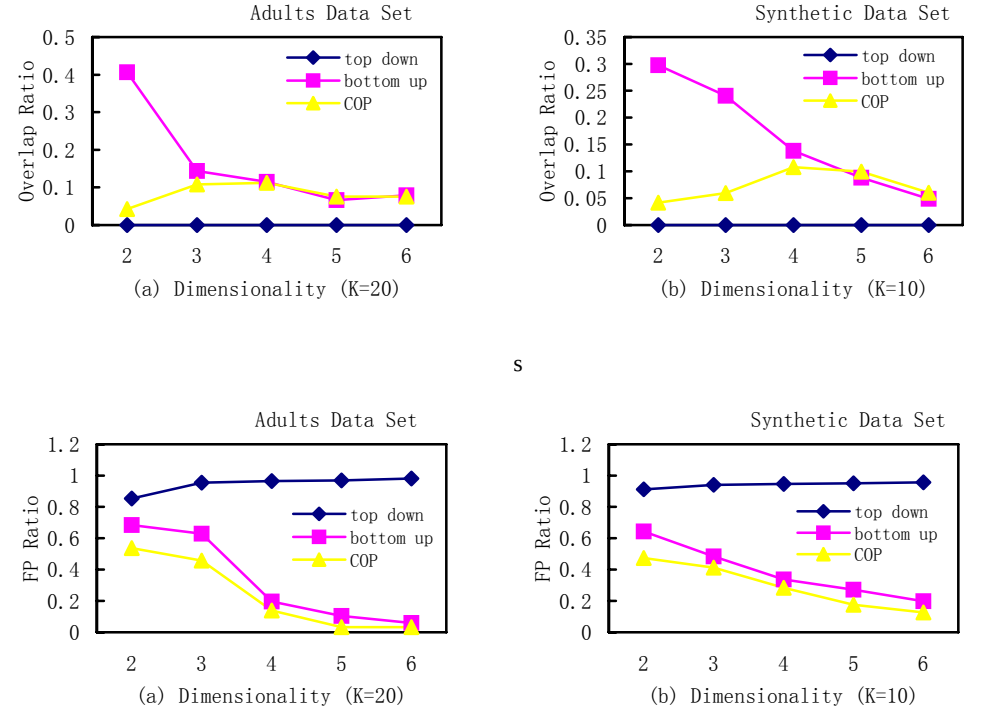


Figure 10. Impacts of K on APS and NPC



s

Figure 11. Comparison of overlap ratio and false positive ratio

In the fourth group, we compare false positive ratio using uniform workload, which is composed of a group of cell like queries covering the whole data space. The results are illustrated in Figure 11(c) and 11(d), from which we can see that with the increasing of dimensionality, false positive ratio of top down partition increases and much larger than the other two partitions. On the other hand, false positive ratio of bottom up partition and COP decreases dramatically with the increasing of dimensionality and COP is always better that bottom up. From calculation of false positive ratio, it can be inferred that it's positively related to the percentage of dead space. When dimensionality increases, the percentage of dead space in top down partition increases while decreases dramatically in the other two partitions, which explains the different tendencies exactly. Besides, the fact that COP has the least false positive ratio come to the conclusion that COP not only performs well in static situation, but also well in dynamic situation.

As for construction time of partition, top down is always the most efficient of all. COP is about 10 times slower than that of top down and 3-4 times slower than that of bottom up. However, since partition is a rarely happened procedure, its performance influence is relatively little and acceptable.

6. CONCLUSION AND FUTURE WORK

In this paper we propose an effective approach (COP)s to form secure multidimensional partition with less efficiency loss. Specifically, considering sparsity and cluster in multidimensional data space, COP first partition space level by level in a top down way, then form optimal partition based on local density clustering. In this paper we limit our partition in static situation. So how to form multidimensional partition in dynamic situation is one important future work. Besides, to support multi-dimensional query, we also need an index to access these secure partitions efficiently, which is another future work.

7. ACKNOWLEDGMENTS

This research is partially supported by two MOE projects under grant number 708004 and 200800020001.

8. REFERENCES

[1] Hacigumus H, Iyer B, Mehrotra S. Providing Database as a Service. In Proc. of 18th International Conference on Data Engineering, San Jose, CA, USA, February 2002.

[2] Li FF, Hadjieleftheriou M, Kollios G, Reyzin L. Dynamic authenticated index structures for outsourced databases. In SIGMOD Conference, pages 121-132. ACM, 2006.

[3] Hacigumus H, Iyer B, Li C, Mehrotra S. Executing SQL over Encrypted Data in the Database Service Provider Model. SIGMOD 2002, June4-6, Madison, Wisconsin, USA

[4] Damiani E, Vimercati SDC, Jajodia S, Paraboschi S, Samarati P. Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs. In 10th ACM CCS, Washington, 2003

[5] Hore, B., Mehrotra, S., Tsudik, G.: A Privacy-Preserving Index for Range Queries. In Proc. of the 30th VLDB Conference, Toronto, Canada, 2004.

[6] Wang JP, Du XY, A Secure Multi-dimensional Partition Based Index in DAS, In Proceeding of the 10th Asia Pacific Web Conference, Shenyang, April 2008, LNCS 4976, pp. 319–330.

[7] Bruno, N., Chauhuri, S., Gravano, L.: STHoles: A Multidimensional Workload-Aware Histogram. In proceedings of the 2001 ACM International Conference on Management of Data (SIGMOD'01), 2001.

[8] LeFevre K., DeWitt D. J., and Ramakrishnan R.. Incognito: Efficient Full-domain k-Anonymity. In Proc. of ACM SIGMOD, pages 49-60, 2005.

[9] LeFevre K., DeWitt D. J., and Ramakrishnan R.. Mondrian Multidimensional k-Anonymity. In Proc. Of ICDE, 2006.

[10] Aggarwal G., Feder T., Kenthapadi K., Khuller S., Panigrahy R., Thomas D., and Zhu A., "Achieving Anonymity via Clustering," in Proc. of ACM PODS, 2006, pp. 153–162.

[11] Meyerson A. and Williams R.. On the Complexity of Optimal K-anonymity. In Proc. of ACM PODS, pages 223-228, 2004

[12] Aggarwal C. C., On k-Anonymity and the Curse of Dimensionality. In Proc. of VLDB, 2005, pp. 901–909.

[13] Xu J., Wang W., Pei J., Wang X., Shi B., and Fu A., "Utility-Based Anonymization Using Local Recoding," in Proc. of SIGKDD, 2006, pp. 20–23.

[14] Agrawal R., Gehrke J., Gunopulos D., Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data, Seattle.

[15] Jagadish HV, Linear Clustering of Objects with Multiple Attributes, Proc. ACM SIGMOD Conf., PP. 332-342, May 1990

[16] Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases. University of Califonia, Ivrine, Dept. of Informaiton and Computer Science, URL= http://mlearn.ics.uci.edu/MLRepository.html.

[17] Ghinita G., Karras P., Kalnis P. and Mamoulis N. Fast Data Anonymization with Low Information Loss. In Proc. of VLDB, 2007, pp. 758-769.