Overlapping Slicing: A Novel Approach for Data Anonymization

K.Sirisha, (M.Tech), Chadalawada Ramanamma Engineering College
Mr.J.Nagamuneiah, Associate Professor, Chadalawada Ramanamma Engineering College

Abstract — publishing data about individuals without revealing sensitive information about them is an important problem. A number of methods have recently been proposed for privacy preserving of multi dimensional records. One of the methods for privacy preserving is data anonymization. Data anonymization techniques, such as generalization, bucketization and slicing have been designed. Generalization cannot be used for high dimensional data. Bucketization cannot separate quasi attributes with sensitive attributes. Slicing provides better privacy but loss of data utility. So, in this paper we introduce a novel technique called overlapping slicing which provides better data utility using *l*- diversity requirement for high dimensional data. We use an efficient algorithm called chi_matrix for attribute correlation. Our experimental evaluation shows that overlapping slicing can have better data utility.

Index Terms—Data anonymization, Data utility, Data Privacy

1 INTRODUCTION

The world is experiencing lot of data collections containing metadata and disk storage apace become increasingly affordable. Metadata is defined as person specific data i.e., information related to one particular person, household or an organization. To protect privacy for these data collections, data anonymization is used. Data anonymization is defined as replacing the contents of identifiable fields in a database. Several data anonymization techniques such as generalization and bucketization differ in the concept of attributes. Attributes are of three categories- 1) identifiers used to identify an individual. E.g.: username, voter id, aadhar card num. 2) Quasi Identifiers whih are public to an individual and when they are linked to the other databases an adversary can get the useful information. E.g.: gender, birth date 3) Sensitive attributes which must be protected and kept in privacy i.e., not known to the adversary. E.g.: occupation, disease, phone number.

Any anonymization technique removes identifiers from the database. Generalization uses k-anonymity procedure to convert the database attributes to higher conceptual data. Generalization partitions the tuples into buckets and then translates the QI values to higher generalized level using k-anonymity. Bucketization partitions tuples into buckets and uses l-diverse procedure in the sensitive attribute column. Generalization works on quasi column. Bucketization works on Sensitive column. Slicing is a high quality technique for anonymization. Slicing is a technique that partitions data both horizontally and vertically.Data anonymization techniques totally partitions the data and permute the sensitive attribute values randomly.

Background

Generalization loses considerable amount of information and not suitable for high dimensional data. Generalization uses k-anonymity principle, it generates tables as – Let RT [A1, A2, ---An] be a table. QI_{RT} be a quasi identifier associated with it. RT is said to be k-anonymity iff each sequence of values in RT [QI_{RT}] appears with at least k-occurrences in RT [QI_{RT}]. If there is no diversity in sensitive attributes values then adversary can attack the database and get knowledge about an individual. If adversary has background knowledge then values can be detected.

Generalization suffers with two types of attacks called - 1) Background Knowledge Attack 2) Homogeneity Attack. K-anonymity can create groups that leak information due to lack of diversity in the sensitive attributes. Generalization uses uniform distribution assumption and correlations between different attributes are lost.

Bucketization provides better data utility than generalization. Bucketization cannot provide separation between quasi and sensitive attributes. Bucketization cannot prevent membership disclosure. Bucketization uses l-diversity principle. A Q* block is l-diverse if a block contains at least "l- well represented values for the sensitive attributes S ". A table is l- diverse if every q* block is l-diverse. Bucketization suffers with two types of attacks. 1) Skewness attack 2) Similarity attack. Skewness attack is defined as overall distribution is skewed by satisfying the l-diversity and does not prevent membership disclosure. Similarity Attack is defined as sensitive attributes in a column are distinct but semantically similar. So, an adversary can learn the important information.

Slicing is a better technique in data anonymization. It partitions data both horizontally and vertically. Slicing preserves membership disclosure and it is suitable for high dimensional data. It cannot provide better data utility for an analyst. Slicing uses l-diversity principle. To overcome all these disadvantages we come up with a novel technique called 'overlapping slicing' which provides better data utility than all data anonymization techniques. It provides better privacy for person specific data.

The rest of this paper is organized as follows-In section 2 we present basic introduction to overlapping slicing, in Section 3 we describe the advantages of

overlapping slicing compared to bucketization, in section 4 we given a brief idea about slicing, in Section 5 an algorithm for overlapping slicing and l-diversified tables and in Section 6 about privacy vs utility, in section 7 describes performance evaluation and in section 8 discussed about conclusion of the paper with some references which are useful to this paper.

2. Overlapping Slicing

Overlapping slicing partitions attributes both horizontally and vertically. Horizontal partitioning is defined as tuples are grouped together. Vertical partitioning is defined as highly correlated attributes are grouped together. This technique provides a clear separation between quasi attributes and sensitive attributes. Sensitive attribute in the relational table should be placed in the each column of a table.

Consider an example as:

Age	Education	Place	Occupation
35	B.Tech	Chennai	PL
38	M.Tech	Mumbai	TL
39	CA	Tirupati	Accountant
45	B.Com	Delhi	CO
46	B.A	Bangalore	I.A.S

Table: 1 Original Database

In the above figure Age, Education, Place are Quasi attributes and occupation is the sensitive attribute which should be not known to the adversary. By applying the overlapping slicing technique on the above table it is formulated as follows.

(Age,Education,Occupation)			(Place, Occupation)	
35	B.Tech	PL	Chennai	PL
38	M.Tech	TL	Mumbai	TL
39	CA	Accountant	Tirupati	Accountant
45	B.Com	CO	Delhi	CO
46	B.A	I.A.S	Hyd	I.A.S

Table: 2 Overlapping sliced Database

In the above figure (Age, education) and (Place, occupation) are highly correlated and formed as columns. But to increase the data utility occupation is added to the first column. First three tuples belong to the first bucket and next two tuples belong to the next bucket.

3. Bucketization vs. Overlapping Slicing

Bucketization is partitioning tuples into bucket and it maintains l-diverse different attributes in the sensitive column. Bucketization uses the algorithm called l-diversity. An equivalence class is said to have l-diversity if there are at least "l –well represented values for the sensitive attribute". A table is said to have l -diversity if every equivalence class of the table has l-diversity.

There are 4 different types of l-diversities.

3.1. Distinct l-diversity. The simplest understanding of "well represented" would be to ensure there are at least l distinct values for the sensitive attribute in each equivalence class. Distinct l-diversity does not prevent probabilistic inference attacks. An equivalence class may have one value appear much more frequently than other values, enabling an adversary to conclude that an entity in the equivalence class is very likely to have that value. This motivated the development of the following two stronger notions of l-diversity.

3.2. Entropy l-diversity. The entropy of an equivalence class E is defined to be Entropy (E) = $\sum p(E,s) \log p(E,s)$ $s \in S$ in which S is the domain of the sensitive attribute, and p(E,s)is the fraction of records in E that have sensitive value s. A table is said to have entropy l-diversity if for every equivalence class E, Entropy(E) $\geq$ log l. Entropy l-diversity is strong than distinct l-diversity. As pointed out in order to have entropy l-diversity for each equivalence class, the entropy of the entire table must be at least log(l). Sometimes this may too restrictive, as the entropy of the entire table may be low if a few values are very common. This leads to the following less conservative notion of l-diversity.

3.3. Recursive (c, l)-diversity. Recursive (c, l)-diversity makes sure that the most frequent value does not appear too frequently, and the less frequent values do not appear too rarely. Let m be the number of values in an equivalence class, and r_i, $1 \leq i \leq m$ be the number of times that the i^{th} most frequent sensitive value appears in an equivalence class E. Then E is said to have recursive (c, l)-diversity if r1< c(rl+rl+1+...+rm).A table is said to have recursive (c, l)-diversity if all of its equivalence classes have recursive (c, l)-diversity

3.4. Probabilistic l- Diversity: An anonymized table satisfies Probabilistic l- Diversity if the frequency of a sensitive value in each group/bucket is at most 1/l. This guarantees that an observer cannot infer the sensitive value of an individual with probability >1/l .l-diverse can have two attacks Skewness Attacks and Similarity Attacks.

Bucketization cannot divide quasi identifiers with sensitive attributes. Bucketization does not prevent membership disclosure. Membership disclosure is termed as disclosing identifiers of an individual. Bucketization reveals data to the adversary as the database is not anonymized. Overlapping slicing provides better data utility with privacy. Overlapping slicing divides quasi attributes with sensitive attributes.

Age	Education	Place	Occupation
35	B.Tech	Chennai	TL
38	M.Tech	Mumbai	PL
39	CA	Tirupati	Accountant
45	B.Com	Delhi	I.A.S
46	B.A	Bangalore	CO

Table: 3Bucketized database

4. Slicing Vs. Overlapping Slicing

Slicing is a data anonymization technique. Slicing prevents membership disclosure. Slicing is used for high dimensional data. Slicing cannot provide better data utility. To overcome this disadvantage we will use over lapping slicing. Overlapping slicing increases better data utility than previous techniques. It provides privacy for person specific data.

(Age, Education)		(Place ,Occupation)	
35	B.Tech	Chennai	PL
38	M.Tech	Mumbai	TL
39	CA	Tirupati	Accountant
45	B.Com	Delhi	CO
46	B.A	Hyd	I.A.S

Table: 4 Sliced Database

Above database describes slicing technique. Data is divided into columns and buckets. Highly correlated attributes are placed in a single column. Tuples are grouped together to form buckets.

5. Overlapping Slicing

Overlapping Slicing can be done in three steps. Attribute Partitioning, Tuple partitioning and column generalization.

5.1Attribute Partitioning

The correlated attributes are partitioned into single column. To find correlation between any two domain values we use three measures

1. Pearson Correlation Coefficient
2. Mean Square Contingency Coefficient.
3. Chi –Square Coefficient.

Mean Square Contingency Coefficient and Chi –Square Coefficient is similar and used for categorical and nominal attributes.
Nominal Attributes: Names of things. It is referred as categorical. Values in it are enumerations.
E.g.: Customer Id, Marital Status, and Occupation.
Discrete attributes: It has a finite or count ably infinite set of values.
Chi square coefficient is a squared deviation of observed and theoritical fequencies. It is used to access two types of comparison. Test Goodness of fit and test of independence.

$$\chi 2 = \sum_{i=1}^{r} \sum_{j=1}^{c} (Oij - Eij)2 \,/Eij$$

E_{ij}=count(A=ai)*count(B=bj)/n

To find the correlated attributes we require to calculate the minimum distance between the attributes.

5.1.1Algorithm Overlapping Slicing: Partition the database into columns by using chi square coefficient
Input: Finite set of tuples with ‘n’ attributes
Output: Highly correlated attributes are generated as columns

1. Consider database DB
   - 1.1 Partition the DB based on chi squared method.
   - 1.2 Find the correlated attributes in the database.
   - 1.3 A1,A2,A3….An be the attributes in the database
   - 1.4 Find correlation between A1 and {A2..An}
     A2 and {A3..An}
     A3 and {A4..An}
     An-1 and{An}
   - 1.5 Highly correlated attributes must be grouped together. (Two attributes per column).
2. Add sensitive attribute to each column in the database.
3. Perform tuple partitioning after attribute partitioning.

To find the correlation between two attributes we need to calculate chi square value for each combination of attributes.Highest valued chi square gives us information about highly correlated attributes. Here we introduce an algorithm called chi_matrix to find the correlated attributes.

5.1.2 Algorithm: chi_matrix finds the correlated attributes
Input: Attributes in the database {A1, A2…….An},
Output: Highly correlated attributes are formed as columns where each column contains two attributes.

1. for each attribute a_i Є A, where i=1 to n-1
2. for each attribute b_j Є A, where j=i+1 to n
3. M (i , j)= chi-square (a_i, b_j)
4. for each row value i Є r, where i=0 to r
5. for each column value j Є c where j=i+1 to c
6. Initialize x to zero and
7. temp(x)=temp(x)U {a[i][j]}
8. pi(x)=pi(x)U{i}
9. pj(x)=pj(x)U{j}
10. Increment x value
11. Where pi and pj are arrays to store position values.
12. Rows=n, columns=n, where n= number of attributes.
13. Convert the matrix representation to single dimensional array named temp.
14. Store row and column wise positions of chi square values in two different types of arrays called pi, pj.
15. Sort the values of temp in descending order and sort row and column positions.
    - 15.1 for each row value i Є x, where i=0 to x
    - 15.2 check the condition if temp(i)>temp(i+1)then
    - 15.3 swap(temp(i),temp(i+1))
    - 15.4 swap(pi(i),pi(i+1))
    - 15.5 swap(pj(i),pj(i+1))
16. Initialize an array ‘h’ to store highly correlated sets.
17. h(0)=h(0) U {temp(0)}
18. hpi(0)= hpi(0)U{pi(0)}
19. hpj(0)=hpj(0)U{pj(0)}
20. for each value k Є x, where k=1 to x
21. for each value m Є k, where m=0 to k
22. if temp(i) is less than or equal to zero
23. then attributes are uncorrelated
24. else, check for common positions in an array pi, pj as pi(k) equals pi(m) and pj(k) equals pj(m)
25. Through flag skip the condition and continue.
26. Else, store the next highest value in the array
27. h(s)=h(s) U {temp(k)}
28. hpi(s)= hpi(s)U{pi(k)}
29. hpj(s)=hpj(s)U{pj(k)},where s is initialized to zero
30. increment s value.
31. Acquire highly correlated attributes in an array called ‘h’.

5.1.3 Algorithm: function chi_square (a_i, b_j)
Input: contingency table CT and two attributes a_i, b_j
Output: Chi square value for two attributes

1. Read observed frequency value for two attributes i, j through contingency table.
2. Calculate the sum of row values 's'
   a1(i) =a1(i) U{s}
3. Calculate the sum of column values 's'
   a2(i) =a2(i) U{s}
4. for each value i Є r , where r = number of rows in CT
5. for each value j Є c, where c = number of columns in CT
6. E (i)(j) = (count (A=ai)*count (B=bj))/n
   Where A, B are attributes,
   ai, bj are distinct values of A, B
7. Calculate chi square value for each combination of attributes.

$$\chi 2 = \sum_{i=1}^{r} \sum_{j=1}^{c} (Oij - Eij)2 \,/Eij$$

8. Return chi square value.

To perform tuple partitioning we need to find distance between two tuples. Tuples with minimum distance are grouped into clusters or buckets. To find the distance we use medoid algorithm.

5.1.4Medoid Algorithm

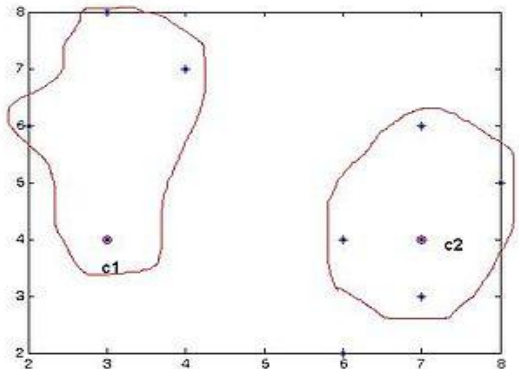
The ***k*-medoid** is a clustering algorithm related to the *k*-means algorithm and the medoid shift algorithm. Both the *k*-means and *k*-medoid algorithms are partitioned (breaking the dataset up into groups) and both attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster.

K-medoid is a classical partitioning technique of clustering that clusters the data set of *n* objects into *k* clusters known *a priori*. A useful tool for determining *k* is the silhouette. The most common realization of *k*-medoid clustering is the **Partitioning Around Medoid (PAM)** algorithm and is as follows:

Algorithm Partitioning Around Medoid (PAM)

Initialize: randomly select *k* of the *n* data points as the medoid

1. Associate each data point to the closest medoid. ("closest" here is defined using any valid distance metric, most commonly Euclidean distance, Manhattan distance or Minkowski distance)
2. For each medoid *m*
   1. For each non-medoid data point *o*
      1. Swap *m* and *o* and compute the total cost of the configuration
3. Select the configuration with the lowest cost
4. Repeat steps 2 to 4 until there is no change in the medoid.



5.2 Tuple Partitioning

Domain values of attributes are called tuples. Partitioning tuples is based on l-diversity algorithm. Each bucket should contain at least l-sensitive values. So the adversary cannot detect sensitive values in the database. To perform tuple partitioning we need to use two basic algorithms as follows-

5.2.1 Algorithm tuple-partition(T, ℓ)

1. Q = {T}; SB = ∅.
2. While Q is not empty
3. Remove the first bucket B from Q; Q = Q − {B}.
4. Split B into two buckets B1 and B2, as in Mondrian.
5. If diversity-check(T, Q ∪ {B1,B2} ∪ SB, ℓ)
6. Q = Q ∪ {B1,B2}.
7. else SB = SB ∪ {B}.
8. return SB.

5.2.2 Algorithm diversity-check (T, T_, ℓ)

1. for each tuple t ∈ T, L[t] = ∅.
2. for each bucket B in T_
3. record f(v) for each column value v in bucket B.
4. for each tuple t ∈ T
5. calculate p(t,B) and find D(t,B).
6. L[t] = L[t] ∪ {p(t,B),D(t,B)i}.
7. for each tuple t ∈ T
8. Calculate p(t, s) for each s based on L[t].
9. if p(t, s) ≥ 1/ℓ, return false.
10. return true.

In the previous algorithm we have used p(t, B) which is defined as probability of occurrence of t sensitive values in bucket B.D(t, B) be the probability of sensitive values s in the distribution. L[t] maintains statistics about one matching bucket B.L[t] is about t's matching buckets. P(t, s) the probability that 't' takes sensitive value 's' calculated upon the "Law of total Probability". In the algorithm for tuple partitioning SB indicates sliced buckets and Q indicates queue containing dataset elements.

5.3 Column Generalization

Column generalization is termed as partitioning attributes into columns and generalizing them for not making known to the adversary. Column generalization is obtained through adding fake tuples into the real database.

If Fake Original Ratio is high then it is more difficult to find the original tuples.

6. Privacy Vs Utility

Privacy loss=Utility Gain

We make correlation between neutral attributes and sensitive attributes. Specific knowledge has a larger impact on privacy. Aggregate information has a larger impact on utility. We cannot have direct comparison between privacy and utility. Privacy is about specific individual. Utility is about aggregate information. Privacy should be enforced for each individual. Utility accumulates all useful knowledge.

False information can cause privacy damage. Correct information contributes to utility gain.
If P_{loss} and U_{loss} will be equal to zero then data utilization will be very high.
\

7. Performance Evaluation

Some preprocessing steps must be applied on the anonymized data before it can be used for workload tasks. In particular, the anonymized table computed by bucketization or slicing contains multiple columns, the linking between which is broken. We need to process such data before workload experiments can run on the data. In bucketization, slicing and overlapping slicing, attributes are partitioned into two or more columns. For a bucket that contains k tuples and c columns, we generate k tuples as follows: We first randomly permutate the values in each column. Then, we generate the i^{th} (1 _ i _ k) tuple by linking the i^{th} value in each column. We apply this procedure to all buckets and generate all of the tuples from all the three anonymized tables. This procedure generates the linking between the two columns in a random fashion. In all of our classification experiments, we apply this procedure 5 times and the average results are reported.

Compare overlapping slicing with bucketization and slicing on data utility of the anonymized data for classifier learning. For all three techniques, we employ the Mondrian algorithm [19] to compute the '-diverse tables. The 'l' value can take values {5, 8, and 10}. Therefore, the sensitive column is always {Place, Occupation}. We evaluate the quality of the anonymized data for classifier learning, which has been used in [11], [12], [3]. We use the Weka software package to evaluate the classification accuracy for Decision Tree C4.5 (J48) and Naive Bayes. Default settings are used in both tasks. For all classification experiments, we use 10-fold cross validation. In our experiments, we choose one attribute as the target attributes (the attribute on which the classifier is built) and all other attributes serve as the predictor attributes. We consider the performances of the anonymization algorithms in both learning the sensitive attribute Occupation and learning a QI attribute Education.

Learning the sensitive attribute- In this experiment, we build a classifier on the sensitive attribute, which is "Occupation." We fix c = 2 here and evaluate the effects of c later in this section. In other words, the target attribute is Occupation and all other attributes are predictor attributes. Fig 1 compares the quality of the anonymized data (generated by the three techniques) with the quality of the original data, when the target attribute is Occupation.

By applying the classification processing on the three data anonymization techniques we can conclude that overlapping slicing can be performed better. Accuracy of the database is being measured through classification in WEKA tool.
Accuracy of bucketization, slicing and overlapping slicing is as follows

		Accuracy (%)
1	Bucketization	20
2	Slicing	25
3	Overlapping slicing	30

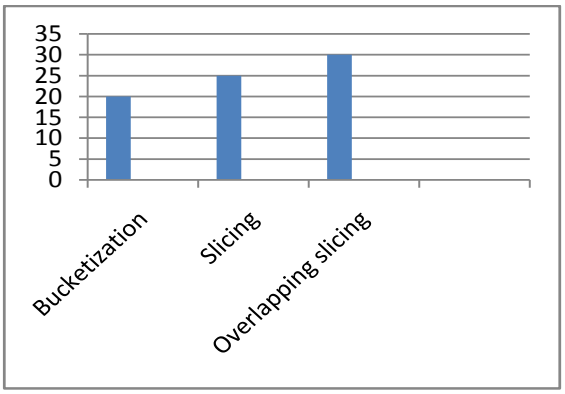


Chart 1

By watching the above graph we need to know the accuracy value has been increased from previous techniques to the present proposed technique. Slicing and overlapping slicing has same accuracy value. While overlapping slicing uses a database with lesser number of attributes but it produces more accuracy than slicing. Overlapping slicing database is as follows

Dataset 1

(Age, Education, Occupation)		
35	B.Tech	PL
38	M.Tech	TL
39	CA	Accountant
45	B.Com	CO
46	B.A	I.A.S

Dataset 2

(Place ,Occupation)	
Chennai	PL
Mumbai	TL
Tirupati	Accountant
Delhi	CO
Hyd	I.A.S

In all experiments, overlapping slicing outperforms both slicing and bucketization, that confirms that slicing pre-serves attribute correlations between the sensitive attribute and some QIs (recall that the sensitive column is {Place , Occupation}). That is mostly due to our preprocessing step that randomly associates the sensitive values to the QI values in each bucket. This may introduce false associations while in generalization, the associations are always correct although the exact associations are hidden.

A final observation is that when 'l' increases, the performances of slicing and bucketization deteriorate much faster than overlapping slicing. This also confirms that overlapping slicing preserves better data utility in work-loads involving the sensitive attribute.

7.1 Results

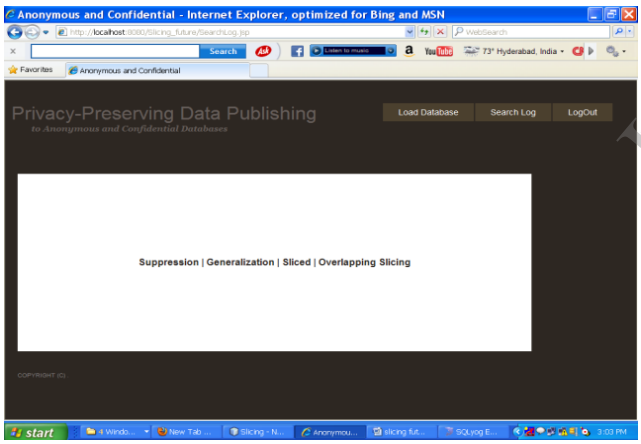


Fig7.1.1: Data Anonymization Techniques

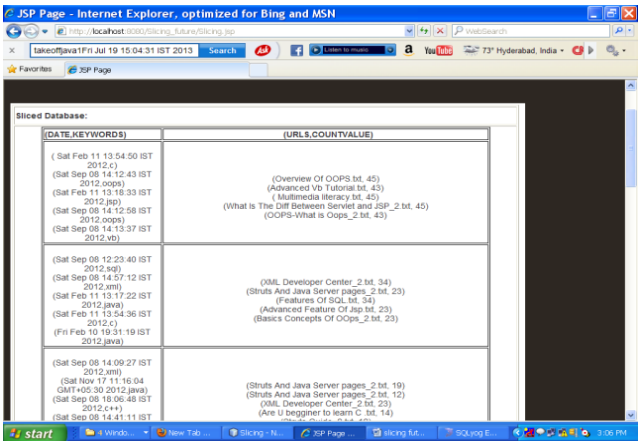


Fig 7.1.2: Sliced Datas

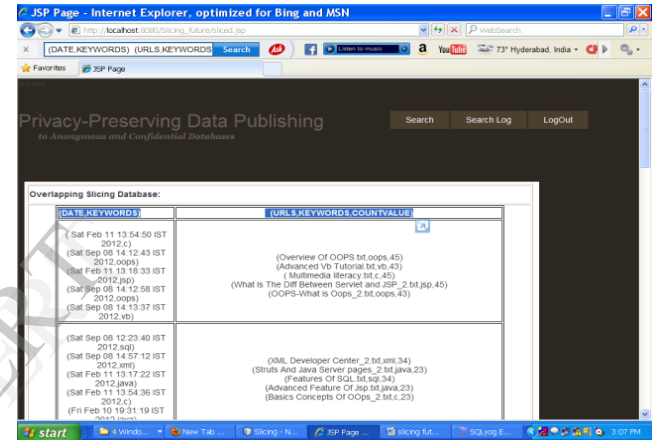


Fig 7.1.3: Overlapping Slicing

8.Conclusion and future work

This paper presents a new approach called overlapping slicing a new approach for data anonymization. Overlapping slicing overcomes the limitations of generalization and bucketization and pre-serves better utility while protecting against privacy threats. We illustrate how to use overlapping slicing to prevent

attribute disclosure and membership disclosure. Our experiments show that slicing preserves better data utility than bucketization and is more effective than slicing.

The general methodology proposed by this work is that: before anonymizing the data, one can analyze the data characteristics and use these characteristics in data anonymization. The rationale is that one can design better data anonymization techniques when we know the data better. We show that attribute correlations can be used for privacy attacks.

This work motivates several directions for future research. First, in this paper, we consider slicing where each attribute is in exactly one column. As an extension we proposed a technique called overlapping slicing, which duplicates an attribute in more than one column. These releases more attribute correlations. For example, in Table 2, one could choose to include

the occupation attribute also in the first column. That is, the two columns are Age; Education; Occupation and Place; Occupation. This could provide better data utility, but the privacy implications need to be carefully studied and understood. It is interesting to study the trade-off between privacy and utility [10].

Second, we plan to study membership disclosure protection in more details. Our experiments show that random grouping is not very effective. We plan to design more effective tuple grouping algorithms.

Third, slicing is a promising technique for handling high-dimensional data. By partitioning attributes into columns, we protect privacy by breaking the association of uncorrelated attributes and preserve data utility by preserving the association between highly correlated attributes.

Finally, while a number of anonymization techniques have been designed, it remains an open problem on how to use the anonymized data. Another direction is to design data mining tasks using the anonymized data computed by various anonymization techniques.

References

[1] Slicing: A New Approach for Privacy Preserving Data PublishingTiancheng Li, Ninghui Li, Senior Member, IEEE, Jian Zhang, Member, IEEE, and Ian Molloy
[2] C. Aggarwal, "On k-Anonymity and the Curse of Dimensionality,"Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 901-909, 2005.
[3] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical Privacy: The SULQ Framework," Proc. ACM Symp. Principles ofDatabase Systems (PODS), pp. 128-138, 2005.
[4] J. Brickell and V. Shmatikov, "The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 70-78, 2008.
[5] B.-C. Chen, K. LeFevre, and R. Ramakrishnan, "Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 770-781, 2007.
[6] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical Privacy: The SULQ Framework," Proc. ACM Symp. Principles of Database Systems (PODS), pp. 128-138, 2005.
[7] J. Brickell and V. Shmatikov, "The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), 70-78, 2008.
[8] B.-C. Chen, K. LeFevre, and R. Ramakrishnan, "Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 770-781, 2007.
[9]H. Cramt'er, Mathematical Methods of Statistics. Princeton Univ. Press, 1948.
[10] Li and N. Li, "On the Tradeoff between Privacy and Utility in Data Publishing," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 517-526, 2009.
[11] B.C.M. Fung, K. Wang, and P.S. Yu, "Top-Down Specialization for Information and Privacy Preservation," Proc. Int'l Conf. Data Eng. (ICDE), pp. 205-216, 2005.
[12] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Workload-Aware Anonymization," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 277-286, 2006.