Integrating Private Databases for Data Analysis*

Ke Wang[1], Benjamin C. M. Fung[1], and Guozhu Dong[2]

[1] Simon Fraser University, BC, Canada
{wangk, bfung}@cs.sfu.ca
[2] Wright State University, OH, USA
gdong@cs.wright.edu

Abstract. In today's globally networked society, there is a dual demand on both information sharing and information protection. A typical scenario is that two parties wish to integrate their private databases to achieve a common goal beneficial to both, provided that their privacy requirements are satisfied. In this paper, we consider the goal of building a classifier over the integrated data while satisfying the k-anonymity privacy requirement. The k-anonymity requirement states that domain values are generalized so that each value of some specified attributes identifies at least k records. The generalization process must not leak more specific information other than the final integrated data. We present a practical and efficient solution to this problem.

1 Introduction

Nowadays, one-stop service has been a trend followed by many competitive business sectors, where a single location provides multiple related services. For example, financial institutions often provide all of daily banking, mortgage, investment, insurance in one location. Behind the scene, this usually involves information sharing among multiple companies. However, a company cannot indiscriminately open up the database to other companies because privacy policies [1] place a limit on information sharing. Consequently, there is a dual demand on information sharing and information protection, driven by trends such as one-stop service, end-to-end integration, outsourcing, simultaneous competition and cooperation, privacy and security.

Consider a concrete scenario. Suppose a bank A and a credit card company B observe different sets of attributes about the same set of individuals identified by the common key SSN, e.g., $T_A(SSN, Age, Balance)$ and $T_B(SSN, Job, Salary)$. These companies want to integrate their data to support better decision making such as loan or card limit approval. However, simply joining T_A and T_B would reveal the sensitive information to the other party. Even if T_A and T_B individually do not contain sensitive information, the integrated data can increase the

* Research was supported in part by a research grant from Emerging Opportunity Fund of IRIS, and a research grant from the Natural Science and Engineering Research Council of Canada.

P. Kantor et al. (Eds.): ISI 2005, LNCS 3495, pp. 171–182, 2005.

Table 1. Compressed tables

Shared		**Party** A		**Party** B		
SSN	**Class**	**Sex**	...	**Job**	**Salary**	...
1-3	0Y3N	M		Janitor	30K	
4-7	0Y4N	M		Mover	32K	
8-12	2Y3N	M		Carpenter	35K	
13-16	3Y1N	F		Technician	37K	
17-22	4Y2N	F		Manager	42K	
23-25	3Y0N	F		Manager	44K	
26-28	3Y0N	M		Accountant	44K	
29-31	3Y0N	F		Accountant	44K	
32-33	2Y0N	M		Lawyer	44K	
34	1Y0N	F		Lawyer	44K	

possibility of inferring such information about individuals. The next example illustrates this point.

Example 1. Consider the data in Table 1 and taxonomy trees in Figure 1. Party A and Party B own

$$T_A(SSN, Sex, \ldots, Class) \text{ and } T_B(SSN, Job, Salary, \ldots, Class)$$

respectively. Each row represents one or more original records and $Class$ contains the distribution of class labels Y and N. After integrating the two tables (by matching the SSN field), the "female lawyer" on (Sex, Job) becomes unique, therefore, vulnerable to be linked to sensitive information such as $Salary$. To protect against such linking, we can generalize *Accountant* and *Lawyer* to *Professional* so that this individual becomes one of many female professionals. No information is lost as far as classification is concerned because $Class$ does not depend on the distinction of *Accountant* and *Lawyer*. ■

In this paper, we consider the following *secure data integration* problem. Given two private tables for the same set of records on different sets of attributes, we want to produce an integrated table on all attributes for release to both parties. The integrated table must satisfy the following two requirements:

Privacy Preservation. The k-anonymity requirement: Given a specified subset of attributes called a "quasi-identifier," each value of the quasi-identifier must identify at least k records. The larger the k, the more difficult it is to identify an individual using the quasi-identifier. This requirement can be satisfied by generalizing domain values into higher level concepts. In addition, at any time in this generalization, no party should learn more detailed information about the other party other than those in the final integrated table. For example, *Lawyer* is more detailed than $Professional$.

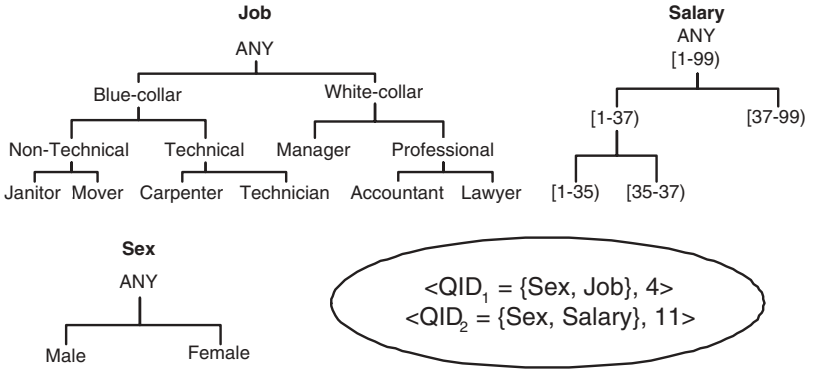

Fig. 1. Taxonomy trees and QIDs

Information Preservation. The generalized data is as useful as possible to classification analysis. Generally speaking, the privacy goal requires masking sensitive information that are *specific* enough to identify individuals, whereas the classification goal requires extracting trends and patterns that are *general* enough to predict new cases. If generalization is "carefully" performed, it is possible to mask identifying information while preserving patterns useful for classification.

There are two obvious approaches. The first one is "integrate-then-generalize": first integrate the two tables and then generalize the integrated table using some single table methods. Unfortunately, this approach does not preserve privacy because any party holding the integrated table will immediately know all private information of both parties. The second approach is "generalize-then-integrate": first generalize each table locally and then integrate the generalized tables. This approach does not work for a quasi-identifier that spans the two tables. In the above example, the k-anonymity on (Sex,Job) cannot be achieved by the k-anonymity on each of Sex and Job separately.

This paper makes two contributions. First, we define the secure data integration problem. The goal is to allow data sharing in the presence of privacy concern. In comparison, classic data integration assumes that all information in private databases can be freely shared, whereas secure multiparty computation allows "result sharing" (e.g., the classifier in our case) but completely prohibits data sharing. More discussions will be available in Section 2. In many applications, being able to access the actual data not only leads to superior results, but also is a necessity. For example, the medical doctor will not trust a given classifier without knowing certain details of patient records. Second, we present a solution to secure data integration where the two parties cooperate to generalize data by exchanging information not more specific than what they agree to share.

2 Related Work

Information integration has been an active area of database research. This literature typically assumes that all information in each database can be freely shared [2]. Secure multiparty computation (SMC), on the other hand, allows sharing of the computed result (i.e., the classifier in our case), but completely prohibits sharing of data [3]. Liang et al. [4] and Agrawal et al. [2] proposed the notion of minimal information sharing for computing queries spanning private databases. They considered computing intersection, intersection size, equijoin and equijoin size. Their model still prohibits the sharing of databases themselves.

The notion of k-anonymity was proposed in [5], and generalization was used to achieve k-anonymity in Datafly system [6] and μ-Argus system [7]. Preserving k-anonymity for classification was studied in [8][9][10]. All these works considered a single data source, therefore, data integration is not an issue. In the case of multiple private databases, joining all databases and applying a single table method would violate the privacy constraint private databases.

3 Problem Definition

We first define k-anonymity and generalization on a single table, then the problem of secure data integration.

3.1 The *k*-Anonymity

Consider a person-specific table $T(D_1, \ldots, D_m, Class)$. The $Class$ column contains class labels or distribution. Each D_i is either a categorical or a continuous attribute. Let $att(v)$ denote the attribute of a value v. The data provider likes to protect against linking an individual to sensitive information through some subset of attributes called a *quasi-identifier*, or QID. A sensitive linking occurs if some value of the QID is shared by only a "small" number of records in T. This requirement is defined below.

Definition 1. Consider p quasi-identifiers $QID_1, \ldots, QID_p$ on T. $a(qid_i)$ denotes the number of records in T that share the value qid_i on QID_i. The *anonymity* of QID_i, denoted $A(QID_i)$, is the smallest $a(qid_i)$ for any value qid_i on QID_i. A table T satisfies the *anonymity requirement* $\{<QID_1, k_1>, \ldots, <QID_p, k_p>\}$ if $A(QID_i) \geq k_i$ for $1 \leq i \leq p$, where k_i is the *anonymity threshold* on QID_i. ■

Note that if QID_j is a subset of QID_i, where $i \neq j$, and if $k_j \leq k_i$, then $<QID_j, k_j>$ is implied by $<QID_i, k_i>$, therefore, can be removed.

Example 2. $<QID_1 = \{Sex, Job\}, 4>$ states that every qid on QID_1 in T must be shared by at least 4 records in T. In Table 1, the following qids violate this requirement: <*M*, *Janitor*>, <*M*, *Accountant*>, <*F*, *Accountant*>, <*M*, *Lawyer*>, <*F*, *Lawyer*>. The example in Figure 1 specifies the k-anonymity requirement on two QIDs. ■

3.2 Generalization and Specialization

To generalize T, a *taxonomy tree* is specified for each categorical attribute in $\cup QID_i$. A leaf node represents a domain value and a parent node represents a less specific value. For a continuous attribute in $\cup QID_i$, a taxonomy tree can be grown at runtime, where each node represents an interval, and each non-leaf node has two child nodes representing some "optimal" binary split of the parent interval. Figure 1 shows a dynamically grown taxonomy tree for *Salary*. More details will be discussed in Section 4.

We generalize a table T by a sequence of specializations starting from the top most value for each attribute. A *specialization*, written $v \rightarrow child(v)$, where $child(v)$ denotes the set of child values of v, replaces the parent value v with the child value that generalizes the domain value in a record. A specialization is *valid* if the specialization results in a table satisfying the anonymity requirement after the specialization. A specialization is *beneficial* if more than one class are involved in the records containing v. A specialization needs to be performed only if it is both valid and beneficial. The specialization process pushes the "cut" of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute D_j, denoted Cut_j, contains exactly one value on each root-to-leaf path. A *solution cut* is $\cup Cut_j$, where D_j is an attribute in $\cup QID_i$, such that the generalized T represented by $\cup Cut_j$ satisfies the anonymity requirement. Figure 2 shows a solution cut indicated by the dashed curve.

3.3 Secure Data Integration

We now consider two parties. Party A owns the table $T_A(ID, D_1, \ldots, D_t, Class)$ and B owns the table $T_B(ID, D_{t+1}, \ldots, D_m, Class)$, over the same set of records. These parties agree to release "minimal information" to form an integrated table T (by matching the ID) for conducting a joint classification analysis. The notion of minimal information is specified by the *joint anonymity requirement* $\{<QID_1, k_1>, \ldots, <QID_p, k_p>\}$ on the integrated table. QID_i is *local* if it contains only attributes from one party, and *global* otherwise.

Definition 2 (Secure Data Integration). Given two private tables T_A and T_B, a joint anonymity requirement $\{<QID_1, k_1>, \ldots, <QID_p, k_p>\}$, and a taxonomy tree for each categorical attribute in $\cup QID_i$, the *secure data integration*

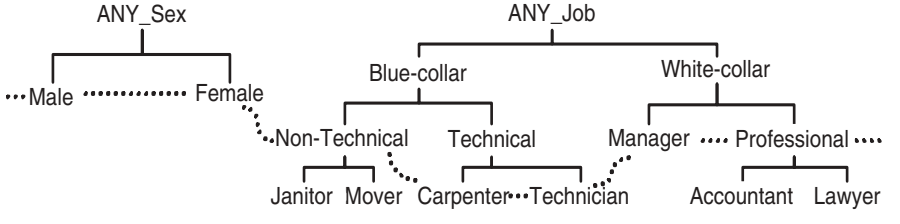

Fig. 2. A solution cut for $QID_1 = \{Sex, Job\}$

is to produce a generalized integrated table T such that (1) T satisfies the joint anonymity requirement, (2) T contains as much information as possible for classification, (3) each party learns nothing about the other party more specific than what is in the final generalized T. ■

For example, if a record in the final T has values F and $Professional$ on Sex and Job, and if Party A learns that $Professional$ in this record comes from $Lawyer$, condition (3) is violated. Our privacy model ensures the anonymity in the final integrated table as well as in any intermediate table.

4 An Unsecured Solution

One unsecured approach is first joining T_A and T_B into a single table T and then generalizing T. Though this approach does not satisfy the requirement (3) in Definition 2 (because the party that generalizes the joint table knows all the details of both T_A and T_B), the integrated table produced satisfies requirements (1) and (2) in Definition 2. Below, we briefly describe this unsecured approach. A secured approach that produces the same integrated table and satisfies the requirement (3) will be presented in Section 5.

A *top-down specialization (TDS)* approach has been proposed in [8] to generalize a single table T. It starts from the top most value for each attribute and iteratively specializes current values until the anonymity requirement is violated. Initially, Cut_i contains the top most value for each attribute D_i. At each iteration, it performs the best specialization w (i.e., of the highest $Score$) among the *candidates* that are valid, beneficial specializations in $\cup Cut_i$, and updates the $Score(x)$ and "valid/benefical" status of x in $\cup Cut_i$ that are affected. The algorithm terminates when there is no more candidate in $\cup Cut_i$.

The core of this approach is computing $Score$, which measures the goodness of a specialization with respect to privacy preservation and information preservation. The effect of a specialization $v \rightarrow child(v)$ can be summarized by "information gain," denoted $InfoGain(v)$, and "anonymity loss," denoted $AnonyLoss(v)$, due to the specialization. *Our selection criterion favors the specialization v that has the maximum information gain per unit of anonymity loss:*

$$Score(v) = \frac{InfoGain(v)}{AnonyLoss(v) + 1}. \quad (1)$$

We add 1 to $AnonyLoss(v)$ to avoid division by zero.

$InfoGain(v)$: Let $T[x]$ denote the set of records in T generalized to the value x. Let $freq(T[x], cls)$ denote the number of records in $T[x]$ having the class cls. Let $|x|$ be the number of elements in a set x. Note that $|T[v]| = \sum_c |T[c]|$, where $c \in child(v)$. We have

$$InfoGain(v) = I(T[v]) - \sum_c \frac{|T[c]|}{|T[v]|} I(T[c]), \quad (2)$$

where $I(T[x])$ is the *entropy* of $T[x]$ [11]:

$$I(T[x]) = -\sum_{cls} \frac{freq(T[x], cls)}{|T[x]|} \times log_2 \frac{freq(T[x], cls)}{|T[x]|}, \quad (3)$$

Intuitively, $I(T[x])$ measures the "mix" of classes for the records in $T[x]$, and $InfoGain(v)$ is the reduction of the mix by specializing v.

$AnonyLoss(v)$: This is the average loss of anonymity by specializing v over all QID_j that contain the attribute of v:

$$AnonyLoss(v) = avg\{A(QID_j) - A_v(QID_j)\}, \quad (4)$$

where $A(QID_j)$ and $A_v(QID_j)$ represents the anonymity before and after specializing v. Note that $AnonyLoss(v)$ not just depends on the attribute of v; it depends on all QID_j that contain the attribute of v.

5 Top-Down Specialization for 2 Parties

Now we consider that the table T is given by two tables T_A and T_B with a common key ID, where Party A holds T_A and Party B holds T_B. At first glance, it seems that the change from one party to two parties is trivial because the change of $Score$ due to specializing a single attribute depends only on that attribute and $Class$, and each party knows about $Class$ and the attributes they have. This observation is wrong because the change of $Score$ involves the change of $A(QID_j)$ that depends on the combination of the attributes in QID_j.

Suppose that, in the TDS approach, each party keeps a copy of the current $\cup Cut_i$ and generalized T, denoted T_g, in addition to the private T_A or T_B. The nature of the top-down approach implies that T_g is more general than the final answer, therefore, does not violate the requirement (3) in Definition 2. At each iteration, the two parties cooperate to perform the same specialization as identified in TDS by communicating certain information in a way that satisfies the requirement (3) in Definition 2. Algorithm 1 describes the procedure at Party A (same for Party B).

First, Party A finds the local best candidate and communicates with Party B to identify the overall winner candidate, say $w \rightarrow child(w)$. To protect the input score, Secure 2-party max [3] can be used. The winner candidate will be the same as identified in TDS because the same selection criterion is used. Suppose that w is local to Party A (otherwise, the discussion below applies to Party B). Party A performs w on its copy of $\cup Cut_i$ and T_g. This means specializing each record $t \in T_g$ containing w into those $t1', \cdots, tk'$ containing child values in $child(w)$, by examining the set of raw records generalized by t, denoted $T_A[t]$, and partitioning $T_A[t]$ among $T_A[t1'], \cdots, T_A[tk']$. Similarly, Party B updates its $\cup Cut_i$ and T_g, and partitions $T_B[t]$ into $T_B[t1'], \cdots, T_B[tk']$. Since Party B does not have the attribute for w, Party A needs to instruct Party B how to partition these records in terms of IDs.

Algorithm 1. TDS2P for Party A

1: initialize T_g to include one record containing top most values;
2: initialize $\cup Cut_i$ to include only top most values;
3: **while** there is some candidate in $\cup Cut_i$ **do**
4: find the local candidate x of highest $Score(x)$;
5: communicate $Score(x)$ with Party B to find the winner;
6: **if** the winner w is local **then**
7: specialize w on T_g;
8: instruct Party B to specialize w;
9: **else**
10: wait for the instruction from Party B;
11: specialize w on T_g using the instruction;
12: **end if**;
13: replace w with $child(w)$ in the local copy of $\cup Cut_i$;
14: update $Score(x)$, the beneficial/valid status for candidates x in $\cup Cut_i$;
15: **end while**;
16: output T_g and $\cup Cut_i$;

Example 3. Consider Table 1 and the joint anonymity requirement:

$\{<QID_1 = \{Sex, Job\}, 4>, \quad <QID_2 = \{Sex, Salary\}, 11>\}$.

Initially,

$T_g = \{<ANY\_Sex,\ ANY\_Job,\ [1\text{-}99)>\}$

and

$\cup Cut_i = \{ANY\_Sex,\ ANY\_Job,\ [1\text{-}99)\}$,

and all specializations in $\cup Cut_i$ are candidates. To find the candidate to specialize, Party A computes $Score(ANY\_Sex)$, and Party B computes $Score(ANY\_Job)$ and $Score([1\text{-}99))$. ■

Below, we describe the key steps: find the winner candidate (Line 4-5), perform the winning specialization (Line 7-11), and update the score and status of candidates (Line 14). For Party A, a *local attribute* refers to an attribute from T_A, and a *local specialization* refers to that of a local attribute.

5.1 Find the Winner Candidate

Party A first finds the local candidate x of highest $Score(x)$, by making use of computed $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$, and then communicates with Party B (using secure 2-party max as in [3]) to find the winner candidate. $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$ come from the update done in the previous iteration or the initialization prior to the first iteration. This step does not access data records. Updating $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$ is considered in Section 5.3.

5.2 Perform the Winner Candidate

Suppose that the winner candidate w is local at Party A (otherwise, replace Party A with Party B). For each record t in T_g containing w, Party A accesses the raw

records in $T_A[t]$ to tell how to specialize t. To facilitate this operation, we represent T_g by the data structure called *Taxonomy Indexed PartitionS (TIPS)*. The idea is to group the raw records in T_A according to their generalized records t in T_g.

Definition 3 (TIPS). TIPS is a tree structure. Each node represents a generalized record over $\cup QID_j$. Each child node represents a specialization of the parent node on exactly one attribute. A leaf node represents a generalized record t in T_g and the *leaf partition* containing the raw records generalized to t, i.e., $T_A[t]$. For a candidate x in $\cup Cut_i$, P_x denotes a leaf partition whose generalized record contains x, and $Link_x$ links up all P_x's. ■

With the TIPS, we can find all raw records generalized to x by following $Link_x$ for a candidate x in $\cup Cut_i$. To ensure that each party has only access to its own raw records, a leaf partition at Party A contains only raw records from T_A and a leaf partition at Party B contains only raw records from T_B. Initially, the TIPS has only the root node representing the most generalized record and all raw records. In each iteration, the two parties cooperate to perform the specialization w by refining the leaf partitions P_w on $Link_w$ in their own TIPS.

Example 4. Continue with Example 3. Initially, TIPS has the root representing the most generalized record <*ANY_Sex, ANY_Job, [1-99)*>, $T_A[root] = T_A$ and $T_B[root] = T_B$. The root is on $Link_{ANY\_Sex}$, $Link_{ANY\_Job}$, and $Link_{[1-99)}$. See the root in Figure 3. The shaded field contains the number of raw records generalized by a node. Suppose that the winning candidate w is *[1-99)* $\rightarrow$ {*[1-37), [37-99)*} on *Salary*.

Party B first creates two child nodes under the root and partitions $T_B[root]$ between them. The root is deleted from all $Link_x$, the child nodes are added to $Link_{[1-37)}$ and $Link_{[37-99)}$, respectively, and both are added to $Link_{ANY\_Job}$ and $Link_{ANY\_Sex}$. Party B then sends the following instruction to Party A:

IDs 1-12 go to the node for *[1-37)*.
IDs 13-34 go to the node for *[37-99)*.

On receiving this instruction, Party A creates the two child nodes under the root in its copy of TIPS and partitions $T_A[root]$ similarly. Suppose that the next winning candidate is $ANY\_Job \rightarrow \{Blue\text{-}collar, White\text{-}collar\}$.

The two parties cooperate to specialize each leaf node on $Link_{ANY\_Job}$ in a similar way, resulting in the TIPS in Figure 4. ■

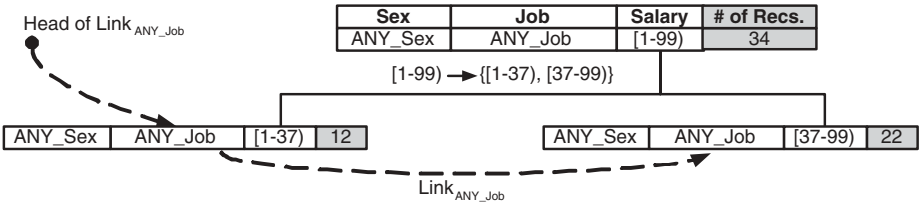

Fig. 3. The TIPS after the first specialization

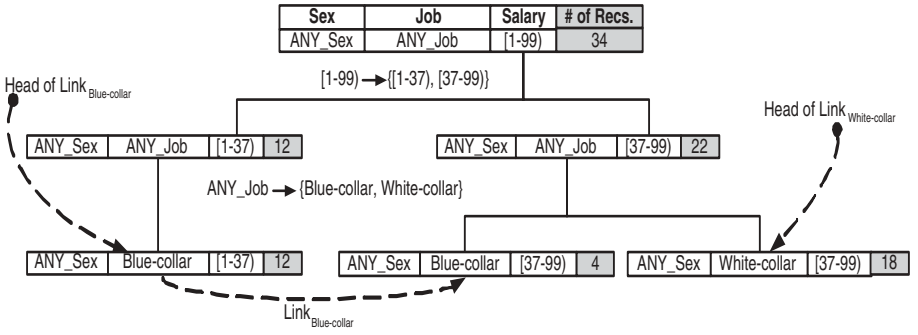

Fig. 4. The TIPS after two specializations

We summarize the operations at the two parties, assuming that the winner w is local at Party A.

Party A**.** Refine each leaf partition P_w on $Link_w$ into child partitions P_c. $Link_c$ is created to link up the new P_c's for the same c. Mark c as "beneficial" if the records on $Link_c$ has more than one class. Also, add P_c to every $Link_x$ other than $Link_w$ to which P_w was previously linked. While scanning the records in P_w, Party A also collects the following information.

- *Instruction for Party B.* If a record in P_w is specialized to a child value c, collect the pair (id,c), where id is the ID of the record. This information will be sent to Party B to refine the corresponding leaf partitions there.
- *Count statistics.* To update $Score$ without accessing raw records, some "count statistics" is maintained for each partition in the TIPS. This is done in the same scan as performing w described above. See the details in [8].

Party B**.** On receiving the instruction from Party A, Party B creates child partitions P_c in its own TIPS. At Party B, P_c's contain raw records from T_B. P_c's are obtained by splitting P_w among P_c's according to the (id, c) pairs received.

We emphasize that updating TIPS is the only operation that accesses raw records. Subsequently, updating $Score(x)$ (in Section 5.3) makes use of the count statistics without accessing raw records anymore.

5.3 Update the Score

$Score(x)$ depends on $InfoGain(x)$, $A_x(QID_j)$ and $A(QID_j)$. The updated $A(QID_j)$ is obtained from $A_w(QID_j)$, where w is the specialization just performed. Below, we consider updating $InfoGain(x)$ and $A_x(QID_j)$ separately.

Updating $InfoGain(x)$**.** $InfoGain(x)$ is affected in that we need to compute $InfoGain(c)$ for newly added c in $child(w)$. The owner party of w can compute $InfoGain(c)$ while collecting the count statistics for c in Section 5.2.

Updating $AnonyLoss(x)$**.** Recall that $A_x(QID_j)$ is the minimum $a(qid_j)$ after specializing x. Therefore, if $att(x)$ and $att(w)$ both occur in some QID_j, the spe-

cialization on w might affect $A_x(QID_j)$, and we need to find the new minimum $a(qid_j)$. The following $QIDTree_j$ data structure indexes $a(qid_j)$ by qid_j.

Definition 4 (QIDTrees). For each $QID_j = \{D_1, \ldots, D_q\}$, $QIDTree_j$ is a tree of q levels, where level $i > 0$ represents generalized values for D_i. A root-to-leaf path represents an existing qid_j on QID_j in the generalized data T_g, with $a(qid_j)$ stored at the leaf node. A branch is trimmed if its $a(qid_j) = 0$. $A(QID_j)$ is the minimum $a(qid_j)$ in $QIDTree_j$. ■

$QIDTree_j$ is kept at a party if the party owns some attributes in QID_j. On specializing the winner w, a party updates its $QIDTree_j$'s that contain the attribute $att(w)$: creates the nodes for the new qid_j's and computes $a(qid_j)$. We can obtain $a(qid_j)$ from the local TIPS: $a(qid_j) = \sum |P_c|$, where P_c is on $Link_c$ and qid_j is the generalized value on QID_j for P_c. $|P_c|$ is part of the count statistics for w collected in Section 5.2.

Example 5. Continue with Example 4. Figure 5 shows the initial $QIDTree_1$ and $QIDTree_2$ for QID_1 and QID_2 on the left. On performing *[1-99)* $\rightarrow$ {*[1-37)*, *[37-99)*}, <*ANY_Sex*, *[1-99)*> in $QIDTree_2$ is replaced with new qids <*ANY_Sex*, *[1-37)*> and <*ANY_Sex*, *[37-99)*>. $A(QID_2) = 12$.

Next, on performing *ANY_Job* $\rightarrow$ {*Blue-collar*, *White-collar*}, <*ANY_Sex, ANY_Job*> in $QIDTree_1$ is replaced with new qids <*ANY_Sex, Blue-collar*> and <*ANY_Sex, White-collar*>. To compute $a(vid)$ for these new qids, we add $|P_{Blue\text{-}collar}|$ on $Link_{Blue\text{-}collar}$ and $|P_{White\text{-}collar}|$ on $Link_{White\text{-}collar}$ (see Figure 4): $a(<ANY\_Sex, Blue\text{-}collar>) = 0 + 12 + 4 = 16$, and $a(<ANY\_Sex, White\text{-}collar>) = 0 + 18 = 18$. So $A_{ANY\_Job}(QID_1) = 16$. ■

Updating $A_x(QID_j)$. This is the same as in [8]. Essentially, it makes use of the count statistics in Section 5.2 to do the update. We omit the details here.

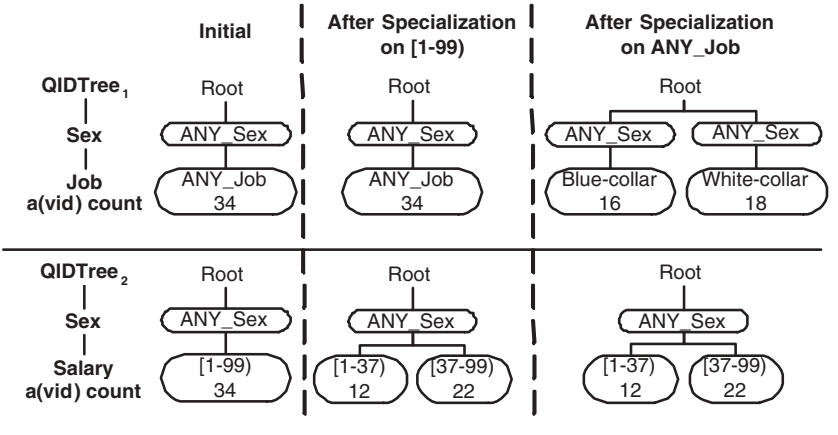

Fig. 5. The QIDTrees data structure

5.4 Analysis

Theorem 1. TDS2P produces exactly the same integrated table as the unsecured TDS on the joint table, and ensures that no party learns more detailed information about the other party other than what they agree to share. ■

This claim follows from the fact that TDS2P performs exactly the same sequence of specializations as in TDS in a distributed manner where T_A and T_B are kept locally at the sources. The only information revealed to each other is those in $\cup Cut_j$ and T_g at each iteration. However, such information is more general than the final integrated table that the two parties agree to share, thanks to the nature of the top-down approach.

We omit the empirical evaluation of the proposed method. Basically, this method produced exactly the same generalized data as in the centralized case where one party holds all attributes of the data (Theorem 1). The latter case has been studied in [8].

References

1. The House of Commons in Canada: The personal information protection and electronic documents act (2000) http://www.privcom.gc.ca/.
2. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California (2003)
3. Yao, A.C.: Protocols for secure computations. In: Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science. (1982)
4. Liang, G., Chawathe, S.S.: Privacy-preserving inter-database operations. In: Proceedings of the 2nd Symposium on Intelligence and Security Informatics. (2004)
5. Dalenius, T.: Finding a needle in a haystack - or identifying anonymous census record. Journal of Official Statistics **2** (1986) 329–336
6. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems **10** (2002) 571–588
7. Hundepool, A., Willenborg, L.: μ- and τ-argus: Software for statistical disclosure control. In: Third International Seminar on Statistical Confidentiality, Bled (1996)
8. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: Proceedings of the 21st IEEE International Conference on Data Engineering, Tokyo, Japan (2005)
9. Wang, K., Yu, P., Chakraborty, S.: Bottom-up generalization: a data mining solution to privacy protection. In: Proceedings of the 4th IEEE International Conference on Data Mining. (2004)
10. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, AB, Canada (2002) 279–288
11. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)