Rule Enforcement with Third Parties in Secure Cooperative Data Access

Meixing Le, Krishna Kant, and Sushil Jajodia

George Mason University, Fairfax, VA 22030
{mlep,kkant,jajodia}@gmu.edu

Abstract. In this paper, we consider the scenario where a set of parties need to cooperate with one another. To safely exchange the information, a set of authorization rules is given to the parties. In some cases, a trusted third party is required to perform the expected operations. Since interactions with the third party can be expensive and there maybe risk of data exposure/misuse, it is important to minimize their use. We formulate the minimization problem and show the problem is in NP-hard. We then propose a greedy algorithm to find close to optimal solutions.

Keywords: Authorization rule, Trusted third party, Join Path.

1 Introduction

In many cases, enterprises need to interact with one another cooperatively to provide rich services. For instance, an e-commerce company needs to obtain data from a shipping company to know the status and cost of a shipping order, and the shipping company requires the order information from the e-commerce company. Furthermore, the e-commerce company may have to exchange data with warehouses and suppliers to get the information about the products. In such an environment, information needs to be exchanged in a controlled way so that the desired business requirements can be met but other private information is never leaked. For example, a shipping company has all the information about its customers. However, only the information about the customers that deal with the e-commerce company in question should be visible to the e-commerce company. The information about the remaining customers should not be released to the e-commerce company. In addition, the data from shipping company may include other information such as which employee is delivering the order, and such information should not be released to the e-commerce company. Therefore, we need a mechanism to define the data access privileges in the cooperative data access environment.

We assume that each enterprise manages its own data and all data is stored in a standard relational form such as BCNF, but it is possible to extend the model to work with other data forms. The data access privileges of the enterprises are regulated by a set of authorization rules. Each authorization rule is defined either on the original tables belonging to an enterprise or over the lossless joins of the data from several different parties. Using join operations, an

L. Wang and B. Shafiq (Eds.): DBSec 2013, LNCS 7964, pp. 282–288, 2013.
© IFIP International Federation for Information Processing 2013

authorization rule only releases the matched information from the parties. For instance, if the e-commerce company can only access the join result of its data and the shipping company's data, then only the tuples about the shipping orders from the e-commerce company can be visible to the e-commerce company. In addition, the attributes such as "delivery_person" are never released to the e-commerce company, so suitable projection operations are applied on the join results in authorization rules to further restrict the access privileges. Hence, the requirement of selective data sharing can be achieved. Selection operations are not considered in the authorization rules.

Under such a scenario, an enterprise may be given an authorization rule on the join result of several relational tables. To obtain the join result, it is required to have one party that has the privileges to access all the basic relations and perform the required join operations. However, due to the access restrictions laid down by the authorization rules, it is possible that no party is capable of receiving all required data. Therefore, we may have to introduce a trusted third party to perform join operations.

Third parties may be expensive to use and the data given to them could be at greater risk of exposure than the data maintained by original parties. In this paper, we focus on the problem of using third parties minimally in order to deliver the information regulated by the given authorization rules. We model the cost of using third party as the amount of data being transferred to the third party, and prove that finding the minimal amount of data to implement a given rule is NP-hard. Therefore, we propose efficient greedy algorithm and evaluate its performance against brute force algorithm. The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 defines the problem and introduces some concepts. Section 4 discusses minimizing the cost of using third parties. Finally, Section 5 concludes the discussion.

2 Related Work

In previous works, researchers proposed models [5] for controlling the cooperative data release. There is also a mechanism [8] to check if an authorization rule can be enforced among cooperative parties. In addition, many classical works discuss query processing in centralized and distributed systems [2,7,3]. However, these works do not deal with constraints from the data owners, which make our problem quite different. There are works such as Sovereign joins [1] to provide third party join services, we can think this as one possible third party service model in our work. Such a service receives encrypted relations from the participating data providers, and sends the encrypted results to the recipients.

Because of the risks associated with third parties, secure multiparty computation (SMC) mechanisms have been developed that ensures no party needs to know about the information of other parties [6,9,4]. However, the generic solution of a SMC problem can be very complicated depending on the input data and does not scale in practice. Therefore, we consider using the third party to implement the rules.

3 Problem Definition and Concepts

We assume the possible join schema is given and all joins are lossless so that a join attribute is always a key attribute of some relations, and only select-project-join queries are considered. An authorization rule denoted as r_t is a triple $[A_t, J_t, P_t]$, where J_t is called the **join path** of the rule which indicates the join over the relational tables, A_t is the authorized attribute set which is the authorized attributes on the join path, and P_t is the party authorized to access the data. For instance, an example rule could be $(R.K, R.X, S.Y), (R \bowtie_{R.K} S) \rightarrow P_t$, where $R.K$ is the key attribute of both R and S, and join path is $R \bowtie S$.

We assume that a trusted third party (TP) is not among the existing cooperative parties and can receive information from any cooperative party. We assume that the TP always performs required operations honestly, and does not leak information to any other party. In our model, we assume the trusted third party works as a service. That is, each time we want to enforce a rule, we need to send all relevant information to the third party, and the third party is only responsible for returning the join results. After that, the third party does not retain any information about the completed join requests. We say an authorization rule can be *enforced* only if there is a way to obtain all the information regulated by the rule. With the existence of a third party, we can always enforce a rule by sending relevant information from cooperative parties to TP. We aim to minimize the amount of data to be sent to the third party.

To find the minimal amount of data to be sent, we can just select rules from the given authorization rules. It is because each rule defines a relational table and we can quantify the amount of information using the data in the tables. We say that a rule is *Relevant* to another if the join path of a rule contains a proper subset of relations of the join path of the other rule. All the rules being selected must be relevant to the target rule denoted as r_t, which is the rule to be enforced. If a relevant rule of r_t is not relevant to any other relevant rules of r_t with longer join paths on the same party, we call it a **Candidate Rule**. We only choose from candidate rules to decide the data that needs to be sent to the TP.

4 Minimizing Cost

In this section, we consider the problem of choosing the proper candidate rules to minimize the amount of information sent to the third party. In our cost model, the amount of information is quantified by sum of the number of attributes picked from each rule multiplied by the number of tuples in that selected rule. Thus, we want to minimize $Cost = \sum_{i=1}^{k} \pi(r_i) * w(r_i)$, where r_i is a selected rule, k is the number of selected rules, and $\pi(r_i)$ is the number of attributes selected to be sent, and $w(r_i)$ is the number of tuples in r_i.

4.1 Rules with Same Number of Tuples

We first assume the candidate rules have the same $w(r_i)$ value. To find the candidate rules that can provide enough information to enforce r_t, we map each attribute in r_t to only one candidate rule so that all of these attributes can be covered. Once we get such a mapping, we have one solution including the selected rules and projections on desired attributes. Among these solutions, we want the minimal cost solution according to our model. Since we assume all the candidate rules have the same number of tuples, it seems that the total cost of each candidate solution should always be the same. However, it is not true because the join attributes appearing in different relations are merged into one attribute in the join results. We can consider the example in Figure 1. The boxes in the figure show the attribute set of the rules, and the join paths and rule numbers are indicated above the boxes. There are four cooperating parties indicated by P_i and one TP, and the three basic relations are joining over the same key attributes $R.K$. Among the 4 candidate rules, if we select r_2, r_3 to retrieve the attributes $R.X$ and $S.Y$ (non-key attributes), we need to send $R.K$ and $S.K$ which are their join attributes to the third party as well. Whereas, if we choose r_1, then we only need to send 3 attributes as $R.K$ and $S.K$ are merged into one attribute in r_1. Thus, choosing a candidate rule with longer join path may reduce the number of attributes actually being transferred. Fewer rules means fewer overlapped join attributes to be sent (e.g., $R.K$ in r_1 and $T.K$ in r_4 are overlapped join attributes). In addition, selecting fewer rules can result in fewer join operations performed at the third party. Since we assume the numbers of tuples in candidate rules are the same, the problem is converted to identify minimal number of candidate rules that can be composed to cover the target attribute set. The problem can be reduced to unweighted set covering problem which is NP-hard.

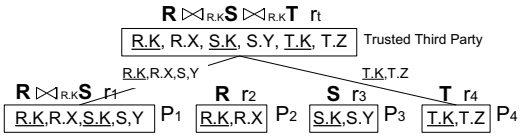

Fig. 1. An example of choosing candidate rules

Theorem 1. *Finding the minimal number of rules sent to the third party to enforce a target rule is NP-hard.*

Proof. Consider a set of elements $U = \{A_1, A_2, ..., A_n\}$ (called the universe), and a set of subsets $S = \{S_1, S_2, ...S_m\}$ where S_i is a set of elements from U. The unweighted set covering problem is to find the minimal number of S_i so that all the elements in U are covered. We can turn it into our rule selection problem. For this we start with the attribute set $\{\underline{A_0}, A_1, A_2, ..., A_n\}$, where A_0

is the key attribute of some relation R and A_i's are non-key attributes of R. For each $S_i \in S$, we construct a candidate rule r_i on R with the attribute set $S_i \bigcup \{A_0\}$ and assign it to a separate cooperative party. Therefore, if we can find the minimal set of rules to enforce some target rule r_t in polynomial time, the set covering problem can also be solved in polynomial time.

4.2 Rules with Different Number of Tuples

In general, the numbers of tuples in the relations/join paths are different, and they depend on the length of the join paths and the join selectivity among the different relations. *Join selectivity* [7] is the ratio of tuples that agree on the join attributes between different relations, and it can be estimated using the historical and statistical data of these relations. In classical query optimization, a large number of works assume such values are known when generating the query plans. We also assume that this is the case. Therefore, we can assign each candidate rule r_i with a cost $cst_i = w(J_i) * \pi(r_i)$, where $\pi(r_i)$ is the per tuple cost of choosing rule r_i, and $w(J_i)$ is the number of tuples in join path J_i. The problem is similar to (but not identical to) the weighted set covering problem. In our problem, once some attributes are covered by previously chosen rules, the following chosen rules should project out these attributes so as to reduce cost. Therefore, our cost function should be as follows where S_i is the attribute set of rule r_i and U is the target attribute set. Basically, the equation says if the key attribute of a rule has already been covered, then one more attribute is added to the cost of choosing this rule.

$$cost(C) = \sum_{i=1}^{k} w(S_i)\pi(S_i), \pi(S_i) = \begin{cases} |S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)|, & \text{if } (key(S_i) \notin \bigcup_{j=1}^{i-1} S_j) \\ |S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)| + 1, & \text{if } (key(S_i) \in \bigcup_{j=1}^{i-1} S_j) \end{cases} \quad (1)$$

Corollary 1. *Finding the minimal amount of information sent to the third party to enforce a target rule is NP-hard.*

Proof. Based on Theorem 1, if we have a polynomial algorithm to find the minimal amount of information with rules of different costs, we can assign the same cost to each candidate rule so as to solve the unweighted version of the problem.

In the weighted set covering problem, the best known greedy algorithm finds the most effective subset by calculating the number of missing attributes it contributes divided by the cost of the subset. In our case, we also want to select the attributes with least costs from the available subsets. Similar to the weighted set covering algorithm which selects the subset S_i using the one with minimal $\frac{w(S_i)}{|S_i \setminus U|}$, we select the rule with the minimal value of $\frac{w(S_i)*\pi(S_i)}{|S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)|}$, where $\pi(S_i)$ is defined in equation (1).

In our problem, with one more rule selected, the third party need to perform one more join operation, and possibly one more join attribute need to be transferred to the third party. Therefore, when selecting a candidate rule, we examine

Algorithm 1. Selecting Minimal Relevant Data For Third Party

Require: The set R of candidate rules of r_t on cooperative parties
Ensure: Find minimal amount of data being sent to TP to enforce r_t

for Each candidate rule $r_i \in R$ **do**
  Do projection on r_i according to the attributes in r_t
  Assign r_i with its estimated number of tuples t_i
The set of selected rules $C \leftarrow \emptyset$
Target attribute set $U \leftarrow$ merged attribute set of r_t
while $U \neq \emptyset$ **do**
  Find a rule $r_i \in R$ that minimize $\alpha = \frac{w(S_i) * \pi(S_i)}{|S_j \bigcap (U \setminus \bigcup_{j=1}^{i-1} S_j)|}$
  $R \leftarrow R \setminus r_i$
  for Each attribute $A_i \in (r_i \bigcap U)$ **do**
    $cost(A_i) \leftarrow w(S_i)$
  $r_i \leftarrow \pi_U(r_i) * w(S_i)$
  $U \leftarrow U \setminus r_i$
  $C \leftarrow C \bigcup r_i$
Return C

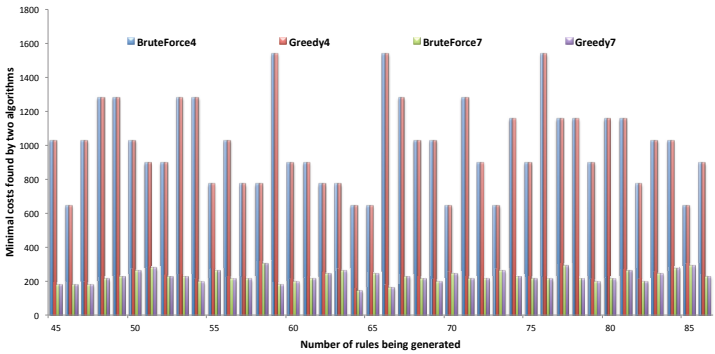

Fig. 2. Minimal communication costs found by two algorithms

the number of attributes this rule can provide and the costs of retrieving these attributes. In the second case of $\pi(S_i)$ in equation (1), the cost of one extra attribute is added. However, if this selected rule can provide many attributes to the uncovered set, the cost of this additional attribute can be amortized. This makes the algorithm prefer rules providing more attributes and results in less number of selected rules which is consistent with our goal. We present our greedy algorithm in Algorithm 1.

We evaluated the effectiveness of our greedy algorithm against brute force algorithm via preliminary simulations. In this simulation evaluation, we use a join schema with 8 parties. The number of tuples in a rule is defined as a function of the join path length, basically $w(J_i) = 1024/2^{length(J_i)-1}$. In other words, we assume as the join path length increases by one, the number of tuples in the results decreases by half. We tested with randomly generated target rules with join path length of 4 and 7. Figure 2 shows the comparison between two algorithms. In fact, the two algorithms generate almost the same results. In Figure 2, the legend of "BruteForce4" indicates the target rule has the join path

length of 4, and brute force algorithm is used. Among these solutions, in less than 2% of the cases the two algorithms produce different answers. In addition, the maximal difference between them is just 5%. The results also indicate the join path length of the target rule affects the costs, but two algorithms give similar solutions independent of the join path length.

5 Conclusions and Future Work

In this paper, we considered a set of authorization rules for cooperative data access among different parties. A trusted third party may be required to do the expected join operations so as to enforce a given rule. We discussed what is the minimal amount of data to be sent to the third party. As the problem is NP-hard, we proposed greedy algorithms to generate solutions which were close to the optimal ones. In the future, we will look into the problem of how to combine the third parties with the existing parties to generate optimal safe query plans.

References

1. Agrawal, R., Asonov, D., Kantarcioglu, M., Li, Y.: Sovereign joins. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, Atlanta, GA, USA, April 3-8, IEEE Computer Society (2006)
2. Bernstein, P.A., Goodman, N., Wong, E., Reeve, C.L., Rothnie Jr., J.B.: Query processing in a system for distributed databases (SDD-1). ACM Transactions on Database Systems 6(4), 602–625 (1981)
3. Chaudhuri, S.: An overview of query optimization in relational systems. In: Proceedings of the 7th ACM Symposium on Principles of Database Systems (1998)
4. Chow, S.S., Lee, J.-H., Subramanian, L.: Two-party computation model for privacy-preserving queries over distributed databases. In: Proceedings of the 16th Network and Distributed System Security Symposium (2009)
5. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Controlled information sharing in collaborative distributed query processing. In: ICDCS 2008, Beijing, China (June 2008)
6. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
7. Kossmann, D.: The state of the art in distributed query processing. ACM Computer Survey 32(4), 422–469 (2000)
8. Le, M., Kant, K., Jajodia, S.: Rule configuration checking in secure cooperative data access. In: SafeConfig 2012 (October 2012)
9. Mishra, D.K., Trivedi, P., Shukla, S.: A glance at secure multiparty computation for privacy preserving data mining. International Journal 1 (2009)