ABSTRACT

The advent of database services has resulted in privacy concerns on the part of the client storing data with third party database service providers. Previous approaches to enabling such a service have been based on data encryption, causing a large overhead in query processing. A distributed architecture for secure database services is proposed as a solution to this problem where data is stored at multiple sites. The distributed architecture provides both privacy as well as fault tolerance to the client. It has been recognized that such distributed systems are vital for the efficient processing required in military as well as commercial applications. For many of these applications it is especially important that the DDBMS should provide multilevel security. For example, the DDBMS should allow users who are cleared at different security levels access to the database consisting of data at a variety of sensitivity levels without compromising security. Security can be applied at multiple levels.

Keywords: fault tolerance, sensitivity level, DDBMS

1. INTRODUCTION

Database service providers are becoming ubiquitous these days. These are companies which have the necessary hardware and software setup (data centers) for storage and retrieval of terabytes of data. As a result of such service providers, parties wanting to store and manage their data may prefer to outsource data to these service providers. The parties who outsource their data will be referred to as clients. The service providers storing data will be referred to as servers. There is a growing concern regarding data privacy among clients. Often, client data has sensitive information which they do not want to compromise. Examples of sensitive databases include a payroll database or a medical database. To capture the notions of privacy in a database, privacy constraints are specified by the client on the columns of the sensitive database.

An example of a privacy constraint is (age, salary) which states that age and salary columns of a tuple must not be accessible together at the servers. The clients also have a set of queries also known as the workload that need to be executed on a regular basis on their outsourced database. Some of the existing solutions for data privacy involve encryption of the entire database when storing it on the server. A client query request requires the entire database to be transferred to the client and decrypted to get the result. However, this has the disadvantage of heavy network traffic as well as decryption cost for each query.

One solution is to encrypt sensitive columns only instead of the entire database so many queries do not require the entire relation to be transmitted to the client. Multiple servers can be used to store the data. The advantage of using two servers is that the columns can be split across the two servers to satisfy privacy constraints without encrypting the split columns.

Thus, in order to satisfy privacy constraints, columns can either be split across servers or stored encrypted. Thus the goal of any decomposition algorithm is to partition the database to satisfy the following.

1. None of the privacy constraints should be violated.
2. For a given workload, minimum number of bytes should be transferred from the servers to the client.

The problem of finding the optimal partition structure for a given set of privacy constraints and query workload can be shown to be intractable. Heuristic search techniques based on Greedy Hill Climbing to come up with nearly optimal solutions.

2. Problem Statement

Given schema R
R (Name, DoB, Gender, ZipCode, Position, Salary, Email, Telephone)
Privacy Constraints: These are described as a collection of subsets of columns of a relation which should not be accessible together. The company may have the following privacy constraints defined:
The problem can now be defined as follows.
We are given:
(1) A data schema R
(2) A set of privacy constraints P over the columns of the schema R
(3) A workload W defined as a set of queries over R.

We have to come up with the best possible decomposition D(R) of the columns of R into R1,R2 and E such that:
(1) All privacy constraints in P are satisfied. These can either be satisfied by encrypting one or more attributes in the constraint or have at least one column of the constraint at each of the servers. Encrypting columns has its disadvantages so priority is given to splitting columns as a way to satisfy privacy constraints.

(2) The cost overhead of D(R) for the workload There should be the minimum possible over all partitions of R which satisfy P. Space is not considered as a constraint and columns of relations are replicated at both servers as long as they satisfy privacy constraints.

Once there is decomposition D(R), given a SQL query posed by the client, query need to be partition into SQL queries for the appropriate servers. The answers to these queries must then be integrated to return the result of the query at the client.

3. System Architecture
The general architecture of a distributed secure database service, is described below. It consists of a trusted client as well as two or more servers that provide a database service. The servers provide reliable content storage and data management but are not trusted by the client to preserve content privacy.

3.1 Data Schema:
This is the schema of the relation the client wishes to store on the server. As a running example, consider a company desiring to store relation R with the following schema.

\[ R \text{ (Name, DoB, Gender, ZipCode, Position, Salary, Email, Telephone)} \]

3.2 Privacy Constraints:
These are described a collection of subsets of columns of a relation which should not be accessible together. The company may have the following privacy constraints defined:

\[ \text{(Telephone)} \cup \text{(Email)} \cup \text{(Name, Salary)} \cup \text{(Name, Position)} \cup \text{(Name, DoB)} \cup \text{(DoB, Gender, ZipCode)} \cup \text{(Position,Salary)} \cup \text{(Salary, DoB)} \]

3.3 Workload:
A workload W is a set of queries that will be executed on a regular basis on the client’s data. A possible workload on R could be:

\[
\begin{align*}
\text{SELECT Name} & \\
\text{FROM R} & \\
\text{WHERE Position = 'Staff';} & \\
\text{SELECT *} & \\
\text{FROM R} & \\
\text{WHERE Salary > 90,000;} & \\
\text{SELECT Name, Email, Telephone} & \\
\text{FROM R} & \\
\text{WHERE Gender = 'F' and ZipCode = '94305';} & \\
\end{align*}
\]

3.4 Tuple ID (TID):
Each tuple of the relation is assigned a unique tuple ID. The TID is used to merge data from multiple servers when executing a query on the data.

3.5 Partitions:
The server consists of two servers to store the client database. The schema and data is partitioned vertically and stored at the two servers. A partition of the schema can be described by three sets $R_1$ (attributes of $R$ stored on Server 1), $R_2$ (attributes of $R$ stored on Server 2) and $E$ (set of encrypted attributes stored on both servers). It is important to note that $(R_1 \cup R_2 \cup E) = R$ and it is not necessarily the case that $R_1 \cap R_2 = \emptyset$. We denote a decomposition of $R$ as $D(R)$.

An example decomposition $D(R)$ of $R$ is given here.

Partition 1 ($R_1$): $(TID, \text{Name}, \text{Email}, \text{Telephone}, \text{Gender}, \text{Salary})$

Partition 2 ($R_2$): $(TID, \text{Position}, \text{DoB}, \text{Email}, \text{Telephone}, \text{ZipCode})$

Encrypted Attributes ($E$): Email, Telephone

4. Query Execution Plans in Distributed Environment

When data is fragmented across multiple servers, there are two plan types used frequently to execute queries on data stored on these servers.

Centralized Plans: On execution of a query, data from each server is transmitted to the client and all further processing is done at the client side. In some cases, multiple requests can go the each server but data from one server is never directly sent over to the other servers.

Consider the following query for an example of centralized query plan Query:

```
SELECT *
FROM R
WHERE Salary > 90,000;
```

In the decomposition $D(R)$, salary is not encrypted. The query is split into the following queries:

**Query 1:**

```
SELECT TID, Name, Email, Telephone, Gender, Salary
FROM R1
WHERE Salary > 90,000
```

**Query 2:**

```
SELECT TID, Position, DoB, ZipCode
FROM R2.
```

In the example, the selection on Salary is pushed to the Server 1. At the client side, a join of the two queries over TID is performed to return the results of the initial query.

Semijoin Plans: As an alternative to centralized plans, it is maybe more efficient to consider semijoin plans. Here, TIDs are passed from one server to the other to reduce the amount of traffic flow to the client. For the same example query, the first query using semijoin plans is:

**Query 1:**

```
SELECT TID, Name, Email, Telephone,
       Gender, Salary
FROM R1
WHERE Salary > 90,000
```

The result of Query 1 is returned to the client and the TIDs are passed to Server 2 to get the matching tuples from R2.

**Query 2:**

```
SELECT TID, Position, DoB,
       ZipCode
FROM R2
WHERE TID in <TIDs returned from Query 1>.
```

The matching tuples from Server 2 are returned back as the results for the query.
5. Conclusion
A distributed architecture for secure database services is proposed as a solution to this problem where data was stored at multiple sites. The distributed architecture provides both privacy as well as fault tolerance to the client. It has been recognized that such distributed systems are vital for the efficient processing required in military as well as commercial applications.

6. References