Security and Integrity Assurance for Join Processing from multiple Data Storages in Cloud

Sharayu M. Anap
Student of ME Computer
MET's IOE, BKC, Nashik,
Savitribai Phule Pune University, Maharashtra, India

K. V. Metre
Asst.Professor
MET's IOE BKC, Nashik,
Savitribai Phule Pune University, Maharashtra, India

ABSTRACT

The Cloud technology is widely used now-a-days. Cloud technology is useful for storing huge amounts of data and also for performing computations. The join queries in cloud are computed by execution server using different storage servers. In the existing system when the client wants to execute a join query, the query is sent to the execution server which gets the data from different storage servers and performs the join operation. The execution server then returns the results to the client. The client must be able to verify the results received from execution server. Various techniques are described in this paper to verify that the results returned by the join query are proper i.e. the results are correct and complete. Also the results provided by the servers must be secure. Integrity indicates that all the valid tuples are included in the result and no valid tuple is missing. Integrity of result helps in further data processing.

Keywords

Join query, Cloud, Integrity.

1. INTRODUCTION

Cloud computing is gaining a lot of importance in today's era. Cloud environment provides easy use of services as required. This technology makes it possible to store huge amount of data and also provides wide services that perform computing. It has separate providers for storing the data and separate providers for performing query executions and return results. Due to this separation of services it is possible to have high availability of data from the storage services and efficient execution of queries. Executing queries in such environments where services are separated has become important in terms of providing security and verifying completeness of the results. It needs to be verified whether the results received after executing the join queries are not tampered or some data is not missing or omitted. Hence it is important to detect if there are any inconsistencies in the data.

2. RELATED WORK

Earlier solutions describe indexing techniques for accessing outsourced data which is secured by encrypting the databases. The queries are executed directly on the encrypted data. For making this possible, indexing information is also stored. That is why, it is necessary to provide data confidentiality and data integrity [1].

While providing database as a service, data is accessed remotely and hence data privacy is of important consideration. Various solutions for data privacy are described which are based on data encryption. It describes the software level and hardware level implementations of the encryption algorithms. In software level encryption, two algorithms i.e. RSA and Blowfish are considered. While storing the data in the database, the data is encrypted and then stored. While retrieving the data, it is decrypted first. The user who encrypts the data uses a key. The database provides the function for encryption and decryption of data to which this key is supplied. Thus only the one who has the key can decrypt and access the data. In the hardware level encryption, special encryption hardware is used [2].

Earlier solutions also describe mechanisms for auditing outsourced data. Integrity auditing is performed to the results sent back by database engine to check if they are correct and complete. Some tuples are added into the outsourced database for auditing the integrity of the data. For inserting such tuples, various approaches are used like the randomized approaches and the deterministic approaches. In randomized approach fake records are generated randomly and are stored at the client side. Randomized approach provides security against attacks, but has to store all the randomly generated fake tuples. In deterministic approach, deterministic functions which are already defined are used from which fake tuples are generated [3].

In case of outsourced databases algorithms have been used for performing join operations. These algorithms use data structure which is authenticated. If the query processing is authenticated then the proper results can be ensured. Performing secure query processing enables user to ensure that all the correct records are included and no records are altered [4].

The use of cloud technology enables the integration of different services that store data and that perform query execution. There are various techniques that can be used to verify the execution of join queries which are assigned to various computing services that cannot be trusted for returning correct results. The user sends a query having join operation and the user may not have any information about where the data is stored and which servers are performing execution of the join queries. The user's request of the join query is sent to the execution server which in turn sends sub-queries to the servers storing data. The results of the sub-queries are then returned to the execution server which then performs the join operation on the two sub-query results and returns the final result to the user [5]. This process is as shown in the Fig. 1. For example, suppose that there are 2 data storage servers, DS1 and DS2 which store two relations A and B. The client can provide query of the form "Select * from A Join B on A.attribute1 = B.attribute1". Here attribute1 is the common attribute on which join will be performed on execution server ES1. After performing the join operation the result R is returned to the client. In this approach, the user has no way to identify if the final result returned by the execution server is correct or not as well as to verify the results returned by the execution server are complete and no data is missing.

Also the results returned to the user by the execution server are not secure and can be tampered.

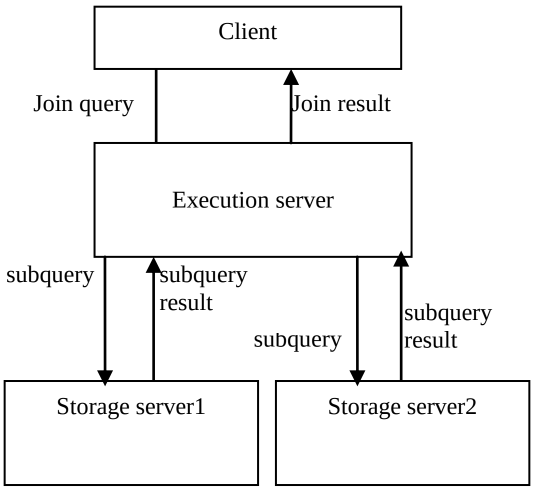

Fig. 1: Existing query evaluation process

Hence the techniques described below provide a way for the user to check if the results returned from the execution server are correct and complete.

3. INTEGRITY TECHNIQUES

The cloud scenario where the join query is to be executed can be considered as having an execution server ES1 and data storage servers DS1 and DS2. Execution server ES1 is responsible for executing the join queries and returns the results to the user. One or more storage servers are used where the data will be stored (DS1 and DS2). The client will request for the join operation and then verify the results returned by the execution server ES1. Before sending the data to the execution server, additional tuples are inserted into the results for verifying integrity at the client side. Various techniques to add such tuples are described below. The storage servers then encrypt the results and return it to the execution server ES1. The subquery results are encrypted to get $R1_{enc}$ and $R2_{enc}$ at the data storage servers DS1 and DS2. The execution server will have the results of two subqueries from the data storage servers. The execution server ES1 performs join operation on the encrypted subquery results $R1_{enc}$ and $R2_{enc}$ and returns the join result which is also encrypted as R_{enc} to the client. The client then decrypts the result of the join query sent by the execution server. Then the client checks for the correctness and completeness of the results. After verifying integrity of the results, the additional tuples are removed and the final results of the join (R) are obtained. After performing integrity check at the client side, the additional tuples are removed to get actual result R. This scenario is depicted in Fig. 2. Using various techniques given below, the data is protected as well as it can be checked if the complete result is returned.

3.1 Use of Cryptographic Techniques

The use of cryptography can be made in order to preserve the information or data stored by the storage servers. Technique called “Encryption on the fly” is used [5]. The data in the database at the storage servers can be encrypted and then sent to the execution server ES1. The execution server can perform operations on the encrypted data and then send the results in the encrypted form to the user. The user can then decrypt the results and then retrieve the original results. If the data is encrypted from more than one storage servers then the encryption key k used for encryption at all the servers must be the same. So that the results would be correct after applying join operation. Since the data is encrypted, it is secure and correct.

3.2 Adding Random Fake Tuples

Tuples with random fake values can be added into the data at the storage servers. Technique known as “Markers” is used [5]. The storage server can add same number of fake tuples as the number of distinct tuples or can get the fake tuples from client. If the number of fake tuples matches the fake tuples at client then it can be concluded that the results are complete. Otherwise the results are considered to be incomplete. So the user can verify whether the same number of fake tuples is present in the results that it had sent to the storage servers. The user can send the number of fake tuples that are to be generated, to the storage servers. As the fake tuples are missing so can the actual tuples also. Hence verifying correct number of fake tuples are present in the result or not can be useful to identify completeness of the join result. However as the size of the database increases, the size of the query result will also increase. In such case adding too many markers will not be effective. Flags can be used to differentiate between the original tuples and the fake tuples.

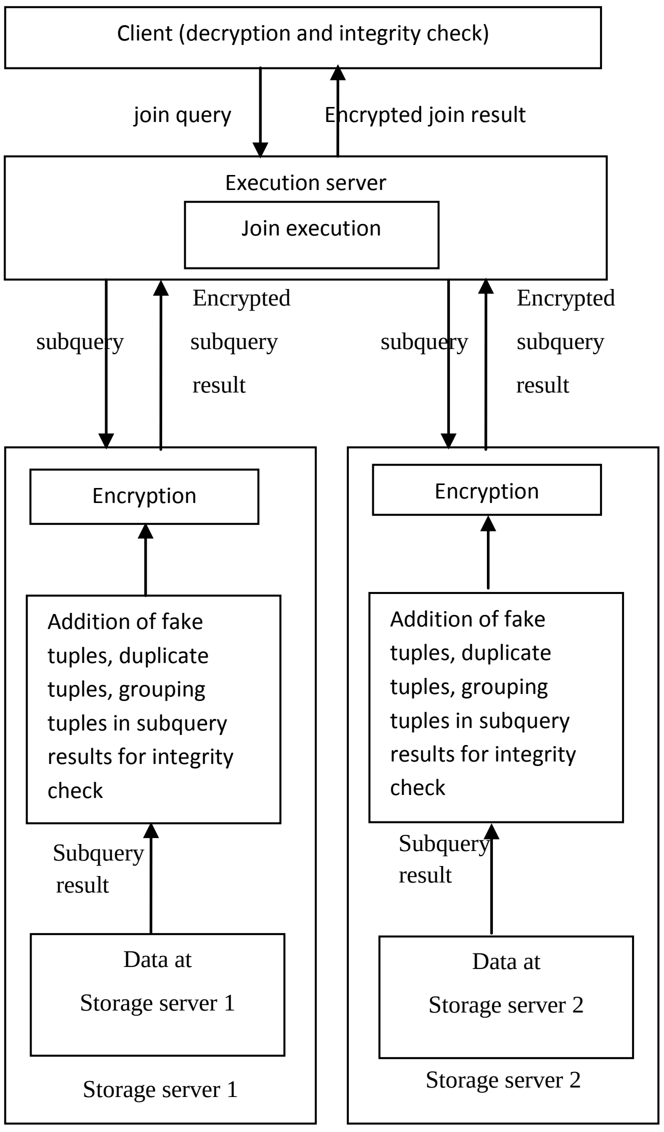
Client (decryption and integrity check)
join query
Encrypted join result
Execution server
Join execution
subquery
Encrypted subquery result
subquery
Encrypted subquery result
Encryption
Encryption
Addition of fake tuples, duplicate tuples, grouping tuples in subquery results for integrity check
Addition of fake tuples, duplicate tuples, grouping tuples in subquery results for integrity check
Subquery result
Subquery result
Data at Storage server 1
Data at Storage server 2
Storage server 1
Storage server 2

Fig. 2: Query evaluation after adding integrity and security techniques

3.3 Using Duplicate Values

Tuples with same values as original values can be added. Technique known as "Twins" is used. Having the tuple as well as its duplicate tuple in the result will ensure that the query result is complete and there has been no data omission. The duplicate tuple can be removed later on to get back the original results. If a tuple appears single (without its duplicate tuple) for which twin tuple is recommended, in the result then it can be ensured that the result is not complete. Providing a twin tuple for every original tuple may provide high integrity. However the size of the database after adding duplicate tuples would be double of the size of the original database. Hence only some tuples can be duplicated based on the condition provided by the client. Another approach for creating duplicate tuples could be using hash function [5].

3.4 Grouping of Tuples

Above techniques can be used for one-to-many join. Tuples with same values can be grouped together into sets. Technique known as "Salts and Buckets" is used [5]. Salts are used to map the different occurrences of the same tuples into different encrypted values. So if the maximum number of occurrences of a tuple in the right hand relation of the join is n then n encrypted tuples with different salt for left hand side relation are created. This technique is generally used in case of one-to-many joins. By using salts the size of the result and the right hand side relation are not change. But the size of the left hand side relation may change. The other approach is called the "Buckets". A bucket is a group of tuples having the same attribute value on which join is to be performed. So if the size of the bucket is n, then a grouping on n tuples is generated which are having same attribute value on which join is to be performed. Dummy values are inserted for the remaining attributes which are not the join attribute. The dummy tuples are added to the right hand side relation and the left hand side relation is not affected. By using buckets, the size of the left hand side relation may not change. But the size of the right hand side relation and the result may change. The salts and buckets are used together.

3.5 Verifying Count of Tuples

The original join query (without addition of fake tuples) from the client can be sent to the execution server to verify the total number of tuples in the result after the join operation. This new query returns only the count of tuples after the join operation. This count, sent by execution server, can be verified against the count of original tuples in the actual results at the client. If both the count matches then it can be concluded that the results are complete. If the count does not match then there is a problem with either results and integrity is not achieved.

4. ALGORITHMIC STRATEGY

1. Get the join query from the client at execution server.
2. Get the fake tuples from the client and send to the storage servers.
3. Send the respective subqueries to the respective storage servers from the execution server.
4. Execute the subqueries at the storage servers to get the subquery results.
5. Insert fake tuples to the subquery results at the storage servers.
6. Encrypt the results and send to execution server.
7. Perform the join on encrypted subquery results at the execution server.
8. Send the encrypted join operation result to the client.
9. At client, decrypt the result and check for fake tuples. If the fake tuples in the result match with the fake tuples at the client then integrity check is done, else return integrity error.
10. Remove the fake tuples and count the total number of original tuples in the result at the client.
11. Get the query from client to verify count.
12. Perform the join on the join attribute and send the count (total number of tuples in result after join operation) to the client.
13. At the client, check if the count received at step 10 matches the count returned by the execution server (in step 12). If the count matches then integrity check is done, else return integrity error.

5. CONCLUSION

The protection and integrity techniques allow the client to verify the confidentiality and correctness and completeness of the results returned by the execution server for join operation. These techniques offer protection and integrity for the data. These techniques are an efficient way for securing the data as well as ensuring that the data is correct and complete. These techniques can be extended to operate on encrypted tables and generating duplicate tuples and dummy tuples for controlling the storage servers. Also these techniques can be used for multiple execution and/or storage servers. These techniques can be applied to parallel joins or chains of joins. These techniques can also be used for investigating different strategies for the join executions and simultaneous execution of the tasks.

6. ACKNOWLEDGMENTS

The authors wish to thank MET's Institute of Engineering Bhujbal Knowledge City Nasik, parents for supporting and motivating for this work because without their blessing this was not possible.

7. REFERENCES

[1] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Modeling and Assessing Inference Exposure in Encrypted Databases," *ACM Trans. Information and System Security*, vol. 8, no. 1, pp. 119-152, Feb. 2005.

[2] H. Hacigu ̈mu ̈s, B. Iyer, and S. Mehrotra, "Providing Database As a Service," *Proc. 18th Int'l Conf. Data Engineering (ICDE '02)*, Feb. 2002

[3] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity Auditing of Outsourced Data," *Proc. 33rd Int'l Conf. Very Large Data Bases(VLDB '07)*, Sept. 2007.

[4] Y. Yang, D. Papadias, S. Papadopoulos, and P. Kalnis, "Authenticated Join Processing in Outsourced Databases," *Proc. ACM Int'l Conf. Management of Data (SIGMOD '09),* June/July. 2009.

[5] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Integrity for Join Queries in the Cloud",*IEEE Trans. Cloud Computing*, vol. 1, no. 2, pp. 187-200, Dec. 2013.