Executing SQL over Encrypted Data in the Database-Service-Provider Model

Summary:
“Software as a service” had become one of the most popular service models in the past. Many organizations were benefited from this service model without spending large amount of money to develop their own software. Due to the success of the model, scientists also wanted to apply the successful experience to database to provide “database as a service.” However, there were different issues need to be resolved before the service is provided to users. One of the most important issue users would concern about is how to assure their data are well-protected. Both users and service providers would worry about the data thefts from outsiders. In this paper, authors proposed an approach to encrypt and decrypt data on client side, and perform most of query processing on service provider side without decrypting data.

Comment:
Personal data could not only be stole from the client sides but also the database service providers. This paper proposed a novel idea to protect user’s data based on the service provider side; it allows user to encrypt and decrypt data mainly on client side and finish most of query process on service provider side which decrease the possibility to have the unencrypted personal data leaked from service providers.

In order to conduct query process for encrypted data, even a simple SQL query could be divided into several queries. For the more complicated query requirements such as joins multiple tables, it demands much more efforts to complete the process. Database maintainers and developers need much additional work to transform existing database management system into this model. In addition, the execution time for query process will increase as well with this model.

In the experiment 1, authors state that the increase of bucket uses to store data will reduce query execution time. Nonetheless, they didn’t provide the detailed information about this experiment. For example, how large the data the dataset it is; how many rows of data should store in each bucket to guarantee a better performance.

Question:
If each row of data is stored into different buckets, could it have the optimized performances? Is there any tradeoff while increasing the numbers of bucket that you could point out?
Human-powered Sorts and Joins

Summary:

Crowdsourcing has become more and more popular recently which is a new service model provides a large group of people to assist customers with various tasks. Amazon’s Mechanical Turk (MTurk) is one of the examples of crowdsourcing. It allows customers to submit their tasks online, and these works can be completed by crowds hired by Amazon. This service model provides the flexibility to customers to find people work for them anytime and anywhere. In this paper, authors demonstrate that crowdsourcing can be applied to conduct two common operations in database management systems, sorting and joining. They also present different optimization methodology such as pre-filtering and task batching to reduce the cost to finish the tasks.

Comment:

In this paper, authors use clear examples to explain how tasks of join and sort operations can be done on MTurk. In addition, they put much effort on establishing different approaches to facilitate these operations. For joining operation, three different user interfaces (simple, naïve batching, and smart batching) are developed and compared in order to figure out the better solution to reduce the cost. For sorting operation, they also introduce three methods (comparison-based, rating-based, and hybrid of these two). Moreover, various experiments are also conducted to verify the performance of different approaches.

According to the approach Authors assign multiple tasks in a batch which is not fair to compare the cost with traditional work (one comparison for each task); especially for the join operation with smart batching. The simple join task authors mention is intuitive for work to complete, but the task with smart batching has become more complicated. In this condition, employees may ask them to increase the unit price for completing tasks in different batches.

The feature filtering optimization technique authors apply to reduce the size of join tasks highly depend on the dataset. Three features they utilize to perform feature filtering in the paper are hair color, skin color, and gender which are only available for identifying people who have these features differently. Since it is difficult to have a general feature filtering approach for various topics, additional work may be required to find out correct features for target subjects. Furthermore, crowdsourcing service provider might charge additional fees for feature filtering since it involves extra computational work.
Question:

The example you give for joining and sorting operation both are simple tasks; however, there are more complicated operations in the real DBMS design. Could you apply the same approach to these complex operations?