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

Summary
This paper explores an algebraic framework to split the SQL query over encrypted data to minimize the computation at the client site in the database-service-provider model. Under the premise that service providers are not trustful, data stored at the database server should be encrypted and may not be decrypted at the provider site. Based on SQL statement features, this paper designed partition functions, identification functions and mapping functions to encrypt data stored into server site database and decryption functions to access decrypt data corresponding to different conditions. This framework provides the database client total privacy, yet continues to benefit from the system-management service of the database service provider.

Pros
The service-provider architecture and the algebraic framework in this paper enable safe storing data at the server side with untrusting providers, and still remain the ability to execute SQL query on the encrypted data without the need to decrypt the stored data. To the author’s experimental evaluation, the computational overhead is not significant.

Cons
For the SQL execution in the server site, the result returned to the client is not always a decided one. For instance, the mapping between original data and the encrypted data in “Attribute = Attribute” conditions could lead to several possibilities, and the client need to further use the metadata to exclude other options to find the result after receiving from the server site.

The result got from the server is encrypted, and decryption computation needs to be done at the client side. It is not truly a cloud computing service to some extent because to my knowledge, almost all computation work should be done on the server site can be count as cloud computing.

For every attribute inside every table, there should be an encryption key, and the “etuple” column also needs another key for encryption. Keys should be stored in the client site as metadata to do the decryption. Therefore, the metadata might be huge if the database is big.

Question
How to evaluate the security it claimed? The author claims that this framework provides client total privacy with their data. How to design experiment to verify or evaluate this claim?

How is the storage expense, time overhead for this system compared with traditional database?
Human-powered Sorts and Joins

Summary
Crowdsourcing marketplace enables to task people with small jobs and provide researchers new directions to perform certain tasks that are computational expensive/unfeasible but easy for human, such as image tagging and classifying. The authors developed a new declarative workflow engine named Qurk to reduce the burden on workflow designers and cost on the task. They concluded and formalized common crowdsourcing tasks as database operations such as filtering, sorting, and joining datasets, and the declarative query processing system Qurk run queries over a crowd of workers, with crowd-based filter, join, and sort operators that optimize for some of the parameters. To reduce the number of HIT, they explored several optimization techniques, such as batching, worker agreement, join pre-filtering, and hybrid sort. In Qurk, they developed different UIs for join comparisons and three approaches for sorting, and their real-world datasets experiment result shows great cost reduce of queries without sacrificing accuracy.

Pros
This work is an innovation because it is the first to systematically study the implementation of operators in a crowdsourced database. They systematically formalized the operations in crowdsourcing activities and abstracted them into a set of database meta-operations, such as join and sort. The abstraction is reasonable, rigorous and easy to map them into real-world operations.

The authors proposed several optimization techniques which combines related tasks together, reduces unnecessary work but still preserving the accuracy. By deploying these different types of optimization techniques, they dramatically reduced the number of HITs needed to perform a crowdsourcing task, and thus reduced the overall cost of all task needed.

Cons
Some crowdsourcing tasks cannot be classified or abstracted into sorts or joins. Other situations should be considered otherwise. For instance, for image tagging task, which needs the crowd to input several words that best describe an image, cannot be counted as either sorts or joins. Another example is Wikipedia, which asks people with expertise in a specific area to complete concepts on their area.

As this paper claims, this work made a dramatic increase on the efficiency for crowdsourcing tasks. However, this work fails to mention the feeling of the crowds. As Qurk take several optimization techniques, this work should conduct study to see whether crowd would be unhappy with the new efficient versions of task (maybe they think the task become more challenging or energy taking that they would be less interested in)

Question
Is there any difference for the feeling of crowds that take tasks on Qurk compared with Murk?