Executing SQL over encrypted data in the database-service-provider model, H. Hacigumus et al., SIGMOD 2002

Summary:

“Database as a service” has been hailed as one of the important milestones in the history of networking and internet technologies. One of the prime concerns in such a case has been data privacy. One concern is related to whether other providers have access to this data, and the other concern has been on the fact that even the provider should also not have access to the entire data, that is it should have data in encrypted form. This paper shows how SQL queries over encrypted data can be used efficiently. The important advantage of this method is that decryption of the data occurs on the client side and hence is more secured from a privacy perspective.

Pros:

1. The idea of a coarse index seems to be very helpful in addressing this issue of security over the network.
2. The splitting of the query into a server and client query is beneficial for cases where the owner of the database does not trust the service provider.

Cons:

1. What if the data size is large, then doing the encryption and decryption might take a long time, this paper does not address the issue of scalability.
2. This work does not talk about logic transformation techniques in detail and that is one area where it could have been improved.

Question:

Is equal interval based partitioning always useful or there should be a model to define such partitions so that the database is optimized?

Human-powered sorts and joins, A. Marcus et al, VLDB 2011

Summary:

With the advent of crowdsourcing applications, Amazon Mechanical Turk (AMT) has gained huge prominence. This paper provides an interesting application of crowdsourcing where humans are made to accomplish a complex task, for sorting and joining data. Moreover, a number of optimizations are also applied that ensures that the costs of running sorts and joins are reduced by a huge factor.

Pros:

1. An effective batching procedure was introduced that reduces complexity of sorting.
2. 3 algorithms to improve the sorting process were presented, more importantly the third algorithm helped understand whether one algorithm complements the other or whether it worsens the performance.

Cons:

1. Baselines other than majority voting can also be used. Why were they not compared against?
2. What if the joint optimization was used in a different way, first sort then join and then first join then sort, how do the results come up?

Question:

Worker selection with questions might not always be fruitful. In that case, can an incremental strategy, where based on the majority answers, Turkers can be provided questions to judge whether they are really spamming or not?