"Executing SQL over Encrypted Data in the Database-Service-Provider Model" by Hakan Hacigumus, Bala Iyer, Chen Li, and Sharad Mehrotra

**Summary:**

In addressing the issue of data privacy in regards to the burgeoning "Database as a Service" model (as part of the emergent Application Service Provider business type), a SQL executing approach is presented that processes as much of a given query on the site provider's side without the need for data decryption and then performing the rest of the task (decryption and query processing) on the client-side. This method is meant to safeguard a corporation's data when handled by a service provider, in effect, tackling the more complex problem of "total" data privacy (where not only are hackers a concern, but the service provider handling the data in the corporation's stead must ensure that the data is protected from itself). An algebraic framework for "query rewriting over encrypted representation," is presented to split the client-side/user query in such a way to as to minimize the amount of client-side computation required while still maintaining the privacy of the user's data on the server side. Using the TPC-H Benchmark, the researchers showed that their proposed architecture did not introduce any significant overhead between the client and the server (in the context of increasing bucket size) and showed that their client-server model approach to query processing performed (time-wise) better than the single-server strategy (which represented the case of trusting the server entirely to encrypt the data). This decrease (with respect to increasing bucket size) in query processing time was also shown in the case of queries made that involved join operations.

**Comments:**

The paper does a great job of building the motivation for the research this group has conducted (though not without flaw, as will be pointed out later in this comment section) and describing the details of their architecture, decryption and mapping mechanisms, and algebraic framework for query splitting. It was interesting to read of the heuristics used in splitting/rewriting a query tree with the aim of minimizing the computation required on the client side of the task (only requiring the client-side for decryption, filtering, and evaluation of the answer to a query). The results found in the experimental sets look promising and show that the system appears to work in satisfying the objective (letting the service provider retain management of client's data while preserving the privacy of that data at minimal computational cost to the client) the researchers set out to achieve. Further, it was important that they found their proposed approach did not create any additional overhead in order to facilitate their communication protocol, which could potentially mitigate the benefits their proposed work offers if the resulting overhead was significant.

The literature survey, found in the introduction, for this paper is quite weak. While there is a brief mention of some related, previous work done with regards to the data privacy issue the researchers attempt to address, the review could be more thorough and perhaps address some of the specifics behind the limitations of previous proposals/studies rather than make a sweeping generalization (that fails to explain why the functionalities of previous system proposals are insufficient for executing complex SQL queries). It would be useful to see a clear addressal of the gaps/specific problems the researchers believe their contribution will overcome, solve, or circumvent. Further, the background/problem context (such as that of data privacy and ASP's) could be further bolstered with some references/studies that support the concern and issue claims the authors make that motivate the need for their work. In addition, while the research group presents their algorithms and frameworks in
detail, they present no formal proof or aggressive, rigorous technique to validate their work. While experimental/empirical validation is useful and important, in the case of security and privacy, a stronger, more rigorous guarantee is needed if the proposed approach/service-provider architecture is to be used in practice.

While the study was conducted in 2002, it would be interesting to see if the results of the experimental evaluation carried over to more modern technology (and exhibited the same performance improvement even though computing technology has gotten faster and equipped with much more Random Access Memory). More importantly, it would be useful to see how this proposed method effectively performs on a more realistic set-up/situation (i.e. using a real server computer, etc.) rather than just in the microcosmic simulation set up for this research study.

Questions:

Would this proposed system carry over to (and perform on) a parallelized server architecture (since hardware and software parallelization optimizations would significantly cut down on the heavier computational costs imposed on the service provider)? Would it, and how if so, need to change (as many computational methods and systems do in the face of parallelization and Amdahl's Law) in order to work on, or better, take advantage of, a parallelized (or distributed) server architecture (given the experiment was conducted on personal computers with single processors)?
"Human-power Sorts and Joins" by Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller

Summary:

The research group describe their design and usage of a declarative query processing engine (Qurk) designed to interface with Amazon's Mechanical Turk crowdsourcing platform and integrate the collective work done by its crowds of human workers to perform high-level database operations, such as join, filter, and sort. Additionally, the researchers describe a variety of optimizations that can be applied to their engine in order to reduce the effective cost of running these "human-powered" sorts and joins (normally re-implemented wastefully by programmers to solve current tasks). It was found, that with batching (which can offer a cost reduction of up to a factor of ten) and feature filtering, the cost of the join task could be reduced from nearly $70 to under $3. In addition, it was shown that batching helped to effectively reduce the complexity associated with the sorting task in both interfaces used in their experimental study, and that their proposed hybrid algorithm, which makes use of aspects of both a comparison-based approach and a rating-based approach (starting with ratings and then use comparisons), performed as well as other sorts but at one third the cost. The study is concluded with complex task involving both joins and sorts, where results show that the research team's approach, in tandem with the various optimizations proposed earlier in the paper, dramatically reduce the total number of HIT's (or Human Intelligence Task, an individual task assigned to a single Turker) from more than 1000 to 77.

Comments:

The paper does a good job explaining in thorough detail the architecture of Qurk and the implementation of its various operations (i.e. how it handles the delegation and design of tasks sent to the Mechanical Turk crowd and aggregation, management, and manipulation of the Mechanical Turk output for given relevant tasks). Furthermore, the various target tasks, join and sort, and their respective optimizations and interface implementations are described in a sufficiently understandable manner (presenting definitions of various metrics used and assuming very little background knowledge on the reader's part). Furthermore, it was interesting to see how the research group's design choices and methods were applied to realistic datasets to evaluate their performance.

While not much appears to be wrong with the work as a whole, it would have been useful to have seen a comparison of these Mechanical-Turk/Qurk-powered tasks (i.e. human-powered tasks) to some automated techniques, such as those from Machine Learning. While the paper did point out certain aspects of tasks (such as the sorting task) where traditional algorithmic approaches would yield unpredictable results or simply fail, there are certain tasks where it would have been interesting to observe how techniques from Machine Vision compared to Qurk-powered method in regards to entity disambiguation. While there is the upfront cost of implementing an algorithmic or automated manner to contend with, once such approaches are implemented, they may perform comparably faster and more accurately than their human-powered counterparts. It would also be interesting to see how machine-learning techniques could be used to further improve the efficiency of tasks such as feature extraction and how this improvement interacts with the costs associated with performing such tasks.
(some Neural Network and clustering-based methods exist for accurately automating the feature extraction process when handling a dataset).

In addition, some of the specific, technical information presented in regards to the optimizations could have been put into an appendix at the end of the work to reduce some clutter right before the experimental results sections were presented (however, this is a minor critique, and as such did not decrease the overall quality of the paper). In regards to the final experiment conducted at the end of the study (i.e. the end-to-end query), while it was useful to observe the promising and dramatic HIT reduction in the context of performing a complex task such as identifying frames in a movie containing actors, it could further benefit this project to show a few more experiments examining other equally complex tasks and confirm this factor of 14.5 HIT reduction statistically. Alternatively, a series of tasks (arranged in increasing complexity) could be arranged to plot the HIT's required (and respective HIT reductions) and observe trends in regards to task complexity and as a means to properly study the scalability of the system.

Questions:

Why was no run-time analysis performed? How does the time it takes the aggregate Qurk/AMT to accomplish a task change with regards to increasing task complexity? What does the Qurk engine in the event than an individual Turker takes too long on a given task (is there a time-out? How does it handle this unpredictable human error case)? It would be interesting to know the run-times/performance times of these human-powered tasks (since any company interested in leveraging human agents would need to consider money as well as time required to accomplish a task/query).