Summary
This paper endeavors to address the challenge of protecting data privacy from the service-providers. It explores techniques to execute SQL queries over encrypted and meta-annotated data. The strategy proposed in the paper is to process as much of the query as possible at the service provider’s site without having to decrypt the data, and decrypt the remainder of query processing at the client site. It presents relation encryption and storage model for data management, and then discusses how a conditional query is translated into a server side query on encrypted data. Particular relational operators are described in details regarding implementation, as well as their experimental results.

Comments
I like this paper for its efforts to explore data protection under the context of complex SQL queries. Even till today, I’ve seen quite a lot researches were invested on this problem, as I believe it is still not completely solved. But it has its merits in several aspects. First, for the purpose of data privacy, the technique proposed in the paper is effective to hide real contents against service providers. Second, it pretty much leverages the processing time between client query and server query for a (limited) set of SQL functionalities. Third, as the translation from client query to server query is fully automatic and behind the interface of SQL database, the framework proposed hereby is friendly and practical to end-users who has no knowledge of its implementation.

However, the technique proposed in this paper has two major flaws. First, it actually only provides SQL functionality over ordered attributes. In building metadata, domain of attributes are mapped into several partitions based on certain order (note random mapping function are applied after the domain partition). Although it is possible to create rule of order for non-structured attribute like address, the order has few clues with functionalities of those attributes. Thus, as an example, it is not possible for one to query addresses that has word “school” without retrieving entire database from server. Second, the orders or categories of attribute is not strictly secured under multiple queries. A possible strategy from server providers to attack the data is to record all historic query from clients and analyze their metadata retrievals. Rows with attributes ranking nearby or within same category are often retrieved at the same time in queries. Therefore, even with random mapping functions, the orders and identities of server-site attribute data are still not securely protected after intense client queries. Thus, justification of the additional complexity and network cost of this paper’s model is very questionable in its own merits. In fact, one can instead pursue a simple affine map of numerical data with module operator or a one-to-many indexes of categories, which from my perspective, as a baseline, is also enough to protect the real contents of data. The experiments shown in the paper were primarily dealing with the reproducibility and efficiency of proposed system, but actually did not justify the privacy and security issues of the encryption database in depth.

Question
To what extent, the proposed encryption SQL database can improve in regards of privacy and security over baseline solution?
Summary
This paper introduces a human-powered database manipulation (sorts and joins) using crowdsourcing marketplaces. It argues pretty amount of tasks in data management are intensively relying on HIT, which are indeed difficult in sense of current machine intelligence. Therefore, the authors built Qurk, a declarative query processing system designed to run queries over a crowd of workers. The system is able to filter, join and sort items via a programmatic interface that combines human computation and traditional relational processing. Specifically it focuses on two operators, joins and sorts, in Qurk. Besides the details of implementation, the authors conduct comparative studies over cost, efficiency and accuracy with its port on MTurk, and eventually demonstrate a level of success.

Comments
Without doubt, it is a conceptually brand novel approaches for database community. The idea is straightforward, but also retain high risks to fail. One prominent contribution of this paper is a formal and serious study approaching HIT efficiency and accuracy under the context of sorts and joins. For sorts, it examines both strategies of rate-based, comparison-based; and for joins, it explores simple join, naive batching and smart batching. It also develops quantitative methods to address some common issues like trade-off between assignments and accuracy of individual worker. As well for the system design, Qurk leverages the efforts to manually setup of crowdsourcing sites, as system designer can quickly login to Qurk and create a database. It is more or less a useful tool.

However, I feel this paper may not fully explore some HIT problems in-depth. First, the reliability of the human powered system depends on how seriously the workers tackles their tasks. It is highly possible for workers to randomly pick answers, as quickly as possible, that might completely messes up results of queries. Therefore I think, the system should have a better mechanism to detect or prevent those unexpected actions. Second, considering various demands and functionalities in crowdsourcing markets, sorts and joins are pretty limited. We human can have much more power than machine. Because human usually are good at performing very complicated, intelligent and multi-objective tasks while a machine only could pursue time-consuming brutal force searches. Asking human to simply emulate a database query is therefore such a simplified way to use human computation.

Question
One particular interesting side functionality of Qurk is to integrate human computation in existing traditional databases, in which small portion of tasks are related to HIT, such as ambiguity removing. Can Qurk pursue its application in such a direction?