Answering Why-Not Questions on Preference Queries (PI: Dr. Lo Chi Lik Eric; 2013/14)

After decades of effort working on database performance, recently the database research community has paid more attention to the issue of *database usability*, i.e., *how to make database systems and database applications more user friendly?* Among all the studies that focus on improving database usability (e.g., SQL query auto-completion), the feature of explaining why some expected tuples are missing in a query result, or the so-called *"why-not?"* feature, is gaining momentum.

A why-not question is being posed to a database when a user wants to know why her expected tuples do not show up in the query result. Currently, end users cannot directly sift through the dataset to determine "why-not?" because the query interface (e.g., web forms) restricts the types of query that they can express. When end users query the data through a database application and ask "why-not?" but do not find any means to get an explanation through the query interface, that would easily cause them to throw up their hands and walk away from the tool forever — the worst result that nobody, especially the database application developers who have spent months to build the database applications, want to see. Unfortunately, supporting the feature of why-not requires deep knowledge of various database query evaluation algorithms, which is beyond the capabilities of most database application developers. In view of this, recently, the database community has started to research techniques to answer why-not questions.

Although a few recent works have discussed techniques for answering why-not questions, they focus on SQL queries and they cannot be used to answer why-not questions on *preference queries*. Faced with information overload, preference queries have been introduced to database systems to present users with the most preferred answers, instead of all the answers. Preference query is useful in

multi-criteria analysis and users would also ask "why-not?" in case their preferred answers are missing in the result. Unfortunately, techniques for answering why-not questions on SQL queries are insufficient for answering why-not questions on preference queries. In a preference query, a tuple can be included in the final result only when it can "beat" many other tuples in the database through tuple-to-tuple comparisons. In contrast, in a SQL query, whether a tuple can be included in the final result mainly depends on whether it can pass through the query predicates, which is independent of most other tuples. The difference between the two answer spaces voids the use of existing techniques to answer why-not questions on preference queries — that signifies that there is a technology gap between what we want to solve (why-not preference query processing) and what we have today (why-not SQL query processing).

In this project, we aim to remove the above technology gap by investigating techniques to answer why-not questions on preference queries. The first challenge is "*what should the answer of a why-not question on preference query (i.e., the explanation) look like?*" One possible explanation type is "explain-by-query-refinement" — teaching the end user how to refine her query such that the missing tuple can go back to the result. The second challenge is: "*what is a good explanation and how to obtain that efficiently (i.e., short running time)?*" We will address this by first understanding the problem complexity and then devising the corresponding algorithms. The third challenge is: "*how to evaluate our proposed solutions for this new kind of problem?*" We will address this challenge by devising an evaluation metric and implementing our proposed solutions as a prototype, and using that for evaluation.

This project will bring advancement to the area of database usability. The project outcome can improve the usability of many database applications.