Letter from the 2016 IEEE TCDE Early Career Award Winner

Chasing Interactivity: Query, *Feedback*, and Result

As we take to data-driven approaches across all disciplines, the ability for practitioners to effectively interact with and make use of the data has become ever more important. While decades of research have gone into making data infrastructure more performant, the focus has typically been on throughput and efficiency of large-scale pipelines, and not the experiences of end users. At the same time, end user data consumption has become pervasive: with over two billion smartphones and tablets sold in the past few years, the number of humans performing complex data tasks is now a significant fraction of the entire human population. We are now at a critical transition point of how the world consumes and interacts with data.

Enabling better *interaction* with the data often solves crucial challenges in search, analysis, and management of data. Consider the example of an exploratory query intent, such as *"I am looking for a particular senior employee's record in the company directory... I'll know it when I see it."* Even when there is a valid result, this query challenges some popular assumptions: that a user is capable of unambiguously expressing their needs, or that there exists a well-formed query to issue to the database. In such cases, providing interactive query interfaces that enable users to quickly sift through and play with the data (e.g., structured autocompletion [1]) can significantly expedite the querying process. In contrast to the traditional *Query→Result* paradigm, we are better served with a new *Query↻Feedback→Result* paradigm. The purpose of *Feedback* is to guide the user to the right query, and is provided in a tightly coupled loop during the query specification step.

While at first glance this seems like a user interface or application-level issue, it is important to note that such a paradigm poses questions to multiple layers of the database stack. At the query language layer, how do we design abstractions that allow iterative composition of queries? From an aggregation and ranking standpoint, how do we provide the right feedback so as to best guide the user? At the execution layer, how do we provide near-instant response times for user feedback? How does the architecture of the overall system change when latency becomes the constraint, as opposed to correctness? We have made attempts to answer some of these questions in the *GestureDB* project [3], which investigates the querying and exploration of databases using touch or motion capture-based gestures. For example, directly mapping gestures to queries would not provide any guidance to the user *during* the articulation of the gesture itself. Thus, we built a gesture recognizer to classify gestures to queries in real time, while using the data and the schema of the database to improve classification quality [4]. To provide perceptibly instantaneous feedback, we used cyclic table scans [2] to surface the first-k tuples of the most likely query result to the end user. While we have been able to flesh out an end-to-end vision for gesture-driven queries, these attempts merely scratch the surface of research in interactive database systems. Answers to these questions will require not only building upon a rich body of existing work in database systems such as iterative query refinement, approximate query processing, and online aggregation, but will additionally require us to draw from a wide variety of complementary disciplines such as visualization, data mining, human-computer interaction, and cognitive psychology.

References

[1] A Nandi, HV Jagadish. *Assisted Querying using Instant-response Interfaces.* SIGMOD 2007.

[2] R Ebenstein, N Kamat, A Nandi. *FluxQuery: An Execution Framework for Highly Interactive Query Workloads.* SIGMOD 2016.

[3] A Nandi. *Querying Without Keyboards.* CIDR 2013.

[4] A Nandi, L Jiang, M Mandel. *Gestural Query Specification.* VLDB 2014.

Arnab Nandi
The Ohio State University