Working Group Report on Responsible and Novel Uses of Data

Chair: Donald Kossmann and Tim Kraska

Traditional database applications are *flat* and *narrow*. The poster-child traditional database application is SAP R/3. It is flat because it sits directly on a relational database system. It is narrow because it requires only a single database to work.

Modern data-intensive applications (including the latest variants of SAP's business application suite) are *deep* and *wide*. They are deep because they consume data that has been produced by a complex data processing pipeline (e.g., data cleaning, data enrichment). They are wide because they consume data from many sources (e.g., millions of sensors and devices, open data, and multiple sources of proprietary data). Applications that make use of IoT and AI technologies are poster-child examples for such deep and wide applications.

There are two characteristics that make the development and operation of wide and deep data-intensive applications challenging:

- *Multi-party*: There are several parties (individual persons and organization) involved in creating and consuming the data. These parties have different interests economically and regarding privacy.
- *Agility*: The system has many moving parts as software components are constantly updated and parties can pull out.

While the semantics of a single SQL database are well understood and standardized, the semantics of a deep and wide pipeline and federation of data management and processing systems are not well understood. As a database community, we need to help all stakeholders of such a complex system. We need to

- provide the right abstractions and tools to specify and reason about the end-to-end cost, security, and utility of such deep and wide data-intensive systems, and
- maintain (and optimize) the properties of such a system as the system changes due to software updates, churning users, changing regulations, and failures.

The industry and scientific community have provided many useful building blocks to address these challenges. It is, however, our job to glue them together and fill in the holes to provide usable and sustainable data management platforms for deep and wide data-intensive applications.

Stakeholders and Research Questions

Traditional (flat and narrow) database applications have three stakeholders: users, application developers, platform providers. Deep and wide database applications expand the set of "user roles": Users can be producers of data, users can be consumers of data, and they can be both, consumers and producers, as part of a step of the deep data processing pipeline. Each of these four roles (consumer, producer, app developer, platform provider) is affected differently by this trend towards deep and wide database applications:

Producers of Data want to control the use of their data. They have an economic and personal interest that the data is only used in the way authorized by the producer. For instance, they might have licensed the use of their personal health records to do medical research, but the might not be comfortable if the results of that medical research is used to develop new medical weapons. In deep and wide data pipelines that aggregate data from many sources and processes these data in multiple steps, it is difficult to reason about the impact of a specific data source. Even though regulations such as GDPR may specify the rights of individuals, it is difficult to enforce these along complex data processing value chains with autonomous parties in each step of the pipeline.

Consumers of Data want to understand the quality of the results that they getting from the data. Flat and narrow data is assumed to be correct, complete, and fresh. Furthermore, flat and narrow data is assumed to be trusted because it was created by the same entity that consumed it as producer and consumer role coincided. In deep and wide database applications, none of these four properties (correctness, completeness, freshness, and trust) can be taken for granted. Consumers need to understand to what extent these properties are fulfilled and reason about the impact of the lack of these properties. For instance, users of a machine translation system need to understand the gender biases created by the system due to the bias in the training data of that system.
Understanding and reasoning about data in such an open data economy becomes particularly difficult as applications become more complex. In the good old days, the semantics of database applications could be formalized using the relational algebra. While it might be possible to reason about the impact of incomplete data on a join operator, we do not have the framework to reason about the impact of incomplete data on the output of, say, a C++ program.

Application developers design, implement, test&debug, and maintain complex applications. Each task of an application developer is heavily impacted by the emergence of deep and wide data processing pipelines. For instance, testing and debugging often require “lab conditions” with reproducible / deterministic behavior. It is likely, however, that many bugs will only be exposed in the wild (i.e., the reality of running the application on real data). How can application developers know that they are working with a sufficiently expressive test data set? How can they simulate the varying availability of data in the agile economy of deep and wide data producers? Furthermore, it is likely that many bugs concern the implementation of the rights of data producers and the worries of data consumers.

The purpose of platform providers is to factor out and automate the needs of all other stakeholders. In an ideal world, the platform would provide the tools that application developers need to deal with the new complexities of deep and wide database applications. Furthermore, the platform would automatically guarantee that the data is not used in ways that are not compliant with the intentions of the data producer and that the data consumer automatically understands and reasons about the quality of the data. To this end, platforms need to provide the right abstractions to specify intentions and needs of all other stakeholders.

Deep and wide data systems involve a complex tradeoff of cost, value, and security of data. For instance, security comes at a cost (e.g., additional processing to encrypt the data or implement integrity protection) and possibly reduces the value of the data (e.g., limits the use of the data). Platforms need to provide the right abstractions and implementation primitives to navigate these tradeoffs.

Summary Recommendations

- We can have the biggest impact by focusing on the end-to-end pipeline to support the producers and consumers of data.
- We should reconsider what a database system means for a deep and wide applications.

Issues to discuss:

- Do deep and wide applications create the hollow middle? How can we build a community which works on such a wide array of topics?
- Many interesting problems exist only when the problem is tackled end-to-end (e.g., a cleaning technique might cause some bias in much later stages). However, building an end-to-end system does require a huge effort, even more than a traditional DB, and no open-source solution exist.
  - Can we collaboratively build a set of open-source tools?
- A lot of the work in new applications domains is extremely hard to publish: papers require an even broader expertise. Furthermore, evaluating those deep and wide approaches is also harder. What can we do about it?
  - Create benchmarks

Things to consider:

- Data Governance from start-to-end: All along the “unclean” to “clean” journey
- Support for debugging both data (input and output) and the pipelines

Gathered Feedback:

Scope: Sub-topics (not exhaustive and based on survey) that should be addressed

- Security, Privacy, Data Governance, and Ethical Use of Data, Bias of samples
- New Applications (IoT, Robotics, Self-Driving Cars)
- Usability and open source