Stop the Truthiness and Just Be Wrong

Oliver Kennedy
University at Buffalo
okennedy@buffalo.edu

Based on discussions with Ying Yang, Niccolò Meneghetti, Poonam Kumari, Will Spoth, Aaron Huber, Arindam Nandi, Boris Glavic, Vinayak Karuppasamy, Dieter Gawlick, Zhen Hua-Liu, Beda Hammerschmidt, Ronny Fehling, Lisa Lu, and many more...

Since their earliest days, databases have held themselves to a strict invariant: Never give the user a wrong answer. So ingrained is it in the psyche of the database community, that those violate have only done so through cumbersome data models [4, 11], huge warning signs [5], or annoying language constructs that break classical SQL [1, 6].

Sadly, by trying to enforce perfection in the database itself, database systems fail to acknowledge that the data being stored is rarely precise, correct, valid, or unambiguous. Emphasizing on certain, deterministic data forces the use of complex, hard-to-manage extract-transform-load pipelines that emit deceptively certain, "truthy" data rather than acknowledging ambiguity or error. As more decisions are automated, even small truthiness errors can drastically impact peoples' lives, from denying a person credit[1], to deciding that an 8-year old is a terrorist[2]. System designers must decide between presenting erroneous data as truthful or risk discarding useful information, and many choose the former.

The database community has already begun treating uncertainty as a first class primitive in databases [8]. Unfortunately, uncertainty also requires us to rethink how we interact with data. For example, personal information managers like Apple Calendar and the iOS Phone App increasingly use facts data-mined from email to automatically populate databases in their contacts and calendar applications. Figure 1 illustrates how these two applications present extracted information to the user. Here, uncertain facts are explicitly kept distinct or clearly marked as being guesses. The interface includes intuitive provenance mechanisms that help to put the extracted information in context. Furthermore, the interface includes overt feedback options to help the user correct or confirm uncertain data. We need to start adapting these techniques to more general data management settings.

The presentation layer isn't the only problem, as identifying sources uncertainty requires developers to invest lots of upfront effort rethinking how they write code. We need to

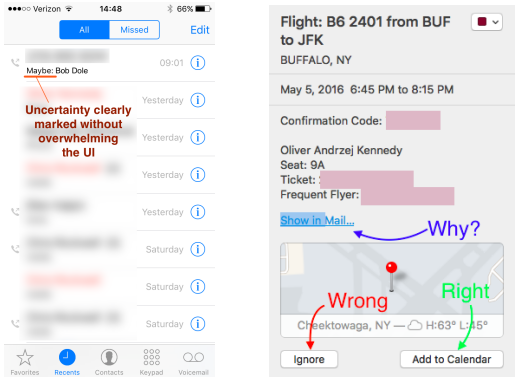

Figure 1: **Extraction in Calendar.app and iOS Phone**

make it worth their while. For example, we might provide infrastructure support to help developers draw generalizations from ambiguous choices [2]. We might streamline imperative language support for uncertainty [9, 10]. Or, we might define higher-order data transformation primitives [3, 12].

In summary, the illusion of accuracy in database query results can no longer be maintained. Database systems must learn how to acknowledge errors in source data, and how to use this information to *effectively* communicate ambiguity to users. Moreover, this needs to happen without overwhelming users, breaking the decades-old abstractions that people understand and use on a day-to-day basis in their workflows, or requiring a statistics background from all users. We need tooling [3, 12] and interfaces [7] that make it easy for databases to be less truthy and more wrong.

[1] http://money.cnn.com/2016/04/11/pf/john-oliver-credit-reports/index.html
[2] http://www.nytimes.com/2010/01/14/nyregion/14watchlist.html?_r=0

1. REFERENCES

[1] L. Antova, C. Koch, and D. Olteanu. $10^{(10^6)}$ worlds and beyond: Efficient representation and processing of incomplete information. *VLDBJ*, 18(5):1021–1040, 2009.
[2] G. Challen, J. A. Ajay, N. DiRienzo, O. Kennedy, A. Maiti, A. Nandugudi, S. Shantharam, J. Shi, G. P. Srinivasa, and L. Ziarek. maybe we should enable more uncertain mobile app programming. In *HotMobile*, 2015.
[3] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: Reliable data cleaning with knowledge bases and crowdsourcing. *pVLDB*, 8(12):1952–1955, Aug. 2015.
[4] A. Deshpande and S. Madden. Mauvedb: Supporting model-based user views in database systems. In *SIGMOD*, 2006.
[5] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *SIGMOD*, 1997.
[6] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. Mcdb: A monte carlo approach to managing uncertain data. In *SIGMOD*, 2008.
[7] P. Kumari, S. Achmiz, and O. Kennedy. Communicating data quality in on-demand curation. In *QDB*, 2016.
[8] D. Suciu, D. Olteanu, C. Ré, and C. Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.
[9] S. Vajda. *Probabilistic programming*. Academic Press, 2014.
[10] S. J. van Schaik, D. Olteanu, and R. Fink. Enframe: A platform for processing probabilistic data. *CoRR*, 2013.
[11] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein. Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1):340–351, 2008.
[12] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy. Lenses: An on-demand approach to ETL. *VLDB*, 8(12):1578–1589, 2015.