The Database Group at the University of Washington

Magdalena Balazinska, Bill Howe, and Dan Suciu
{magda,billhowe,suciu}@cs.washington.edu
Dept. of Computer Science & Engineering
University of Washington

The database group at the University of Washington (UW) was founded in 1998 when the department hired Alon Halevy (now at Google). The group currently consists of about twenty researchers: three faculty members (the authors), four postdocs, and fifteen students. Alumni include faculty members at Computer Science Departments at British Columbia, Michigan, Pennsylvania, Stanford, UMass, Wisconsin, one faculty member at the CMU Tepper School of Business, and several researchers and engineers at Facebook, Google, Microsoft, Nokia, Twitter, and other technology companies. The group has funding from NSF, the Gordon and Betty Moore Foundation, the Alfred P. Sloan Foundation, and several companies including Amazon, EMC, Google, HP, Intel, Microsoft, NEC, and Yahoo. The group has been recognized through several best paper awards and two ACM SIGMOD Best Dissertation Awards.

We conduct research mostly in small groups and tackle a diverse set of data management challenges. Some of our projects result from collaborations with domain scientists on the UW campus; others are sparked by novel theoretical breakthroughs that lead to new approaches to data management challenges; many are the results of both. We give here a short overview of the recent research themes in our group; more details are available on our website:

`http://db.cs.washington.edu/`

1. SCIENTIFIC DATA MANAGEMENT

Our research agenda is partially derived from collaborations with scientists across the University of Washington and beyond, leveraging our close connection with the University of Washington eScience Institute [6].

The eScience Institute was founded in 2005 with the goal of advancing the research and practice of data-intensive discovery across all fields of science. With the advent of new, high-bandwidth data sources (survey telescopes, high-throughput sequencers, ubiquitous sensor networks, planetary-scale simulations), data management research became recognized as a critical driver of scientific discovery. As a result, the database group and the eScience Institute became close partners, and were able to initiate and maintain multiple long-term collaborations with scientists.

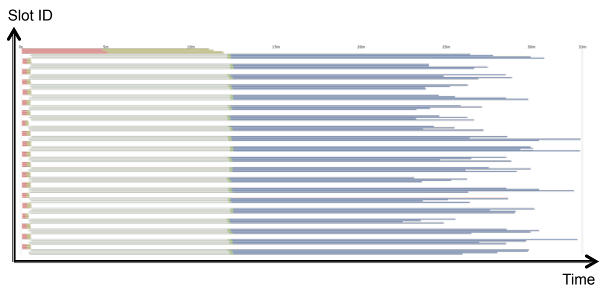
In 2008, we founded an inter-disciplinary research group called AstroDB [1]. This group brings together faculty, research scientists, postdocs, and students from the Astronomy department and our database group. In 2009, we initiated an independent collaboration with a marine microbiology lab. Thanks to the sustained nature of these partnerships, both have led to a series of joint research projects. We give examples in the following sections.

Our inter-disciplinary collaborations have also allowed us to collect a curated repository of datasets and use cases that anyone can use in their research: A repository of MapReduce applications [15], a public repository of scientific datasets equipped with a SQL interface [19], and a number of parallel analytics use cases that go beyond MapReduce [14]. We are continuously working on expanding these collections of applications.

2. BIG DATA SYSTEMS

Motivated by partnerships in both science and industry, we have invested deeply in research on systems that can empower non-specialists to extract knowledge from large, complex, and noisy datasets.

As one approach, we worked on leveraging modern tools such as Dryad and Hadoop (see our Nuage project website for details and papers [15]). As an example, we helped our AstroDB collaborators to develop a method for carrying out an important analysis step, which involved data clustering, using both Dryad and Hadoop. We found, however, that it was far from trivial to use these systems in a way that resulted in high performance: while both Dryad and Hadoop made it easy to express the user-defined functions that we needed, both lead to problems with uneven load distribution, also called *skew*. This observation lead us to build the SkewReduce and SkewTune [13] systems; both are publicly available and have been very well received by sci-



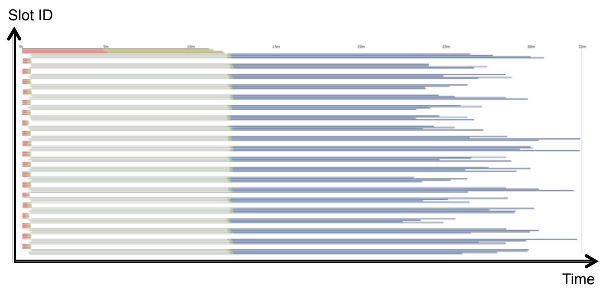


Figure 1: Timing chart illustrating how SkewTune can help squeeze the execution of a MapReduce job. In this example, runtime decreases by 38% with SkewTune.

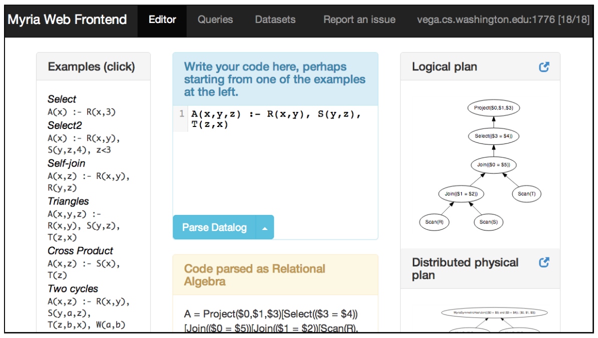
ence and industry. Figure 1 illustrates the benefits of SkewTune on the execution of a MapReduce job. In that same line of work, we developed progress indicators for MapReduce workflows and a fault-tolerance optimizer. These approaches were designed to facilitate and accelerate data analysis using Hadoop.

Simultaneously, we became interested in extending existing Big Data systems with iterative capabilities to support more general classes of complex analytics tasks. The HaLoop system extended Hadoop with inter-iteration caching that led to orders of magnitude performance improvements for long-running iterative jobs [3]. We followed up that work by showing how to implement and optimize a recursive query language — Datalog — using the HaLoop runtime.

Our work became quickly noted in the community: the HaLoop paper is currently the most-cited paper in VLDB 2010, SkewTune and one of our MapReduce progress estimation papers are among the top cited in SIGMOD 2012 and ICDE 2010[1], while a related paper on the study of skew in MapReduce won the Best Student Paper at the Open Cirrus Summit 2011.

Over the past two years, our group has moved beyond Hadoop. We have built our own, parallel data management system called Myria [14]. An important aspect of our system is that Myria is set up as a Cloud service that users access directly from their browsers. In the Myria project, we are studying *both* the systems challenges related to efficiently supporting modern data management and analytics needs *and* what it means to operate such as system as a Cloud service. Myria now runs on 100-node Amazon EC2 deployments and processes terabytes of data from applications in astronomy, oceanography, social media, and cybersecurity, as well as standard benchmarks. Figure 2 shows a screenshot of the Myria graphical interface.

Myria's goal is to allow users to simply upload their data and become self-sufficient data science experts: "self-serve analytics." Myria accepts queries (online in a



Write your code here, perhaps starting from one of the examples at the left.

```
A(x,y,z) :- R(x,y), S(y,z),
T(z,x)
```

Select
```
A(x) :- R(x,3)
```
Select2
```
A(x) :- R(x,y),
S(y,z,4), z<3
```
Self-join
```
A(x,z) :- R(x,y),
R(y,z)
```
Triangles
```
A(x,y,z) :-
R(x,y), S(y,z),
T(z,x)
```
Cross Product
```
A(x,z) :- S(x),
T(z)
```
Two cycles
```
A(x,z) :- R(x,y),
S(y,a,z),
T(z,b,x), W(a,b)
```

Code parsed as Relational Algebra

A = Project($0,$1,$3)[Select(($3 = $4))
[Join(($0 = $5))[Join(($1 = $2))[Scan(R),

Figure 2: Myria browser-based front-end.

browser or programmtically through a REST API) written in SQL, Datalog, and a new, iterative, hybrid declarative/imperative language we call MyriaL. All three languages are compiled to the same intermediate representation based on an extension of RA+While, then optimized to produce a parallel physical plan for execution on a cluster. Based on our experience with SQLShare [9] (described below), we know that science users can and will write data analysis tasks in declarative languages, but we seek new language features to capture a greater proportion of their tasks. Myria's execution layer, MyriaX, adopts state-of-the-art system design principles: it uses a pipelined, possibly cyclic graph of dataflow operators that make efficient use of I/O and memory, and it has built-in support for asynchronous evaluation of recursive queries. In addition, we are innovating in the space of massively distributed query processing techniques, by building on recent theoretical breakthroughs connecting conjunctive queries with the fractional edge cover (for sequential processing) or fractional edge packing (for parallel processing). The suite a techniques lead to significant reductions in the amount of data communication during the computation of multiway join queries; some of our work in this space is here [2], more is under way. Figure 3 shows the high-level Myria architecture. We will be demonstrating Myria at SIGMOD 2014.

Separately from Myria, we are also working on a

[1]`http://arnetminer.org/conferencebestpapers`

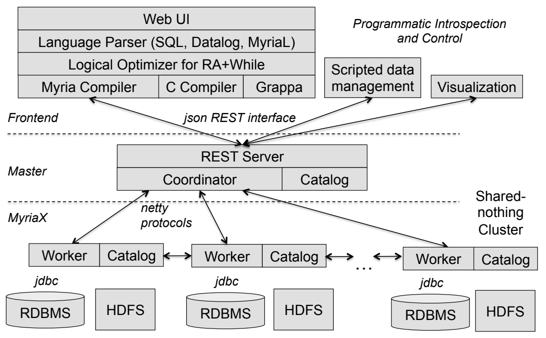


Figure 3: Myria system architecture.

project to manage non-relational data in the context of scientific data management. In astronomy, for example, research involves the analysis of telescope images, such as those produced by the Sloan Digital Sky Survey or the Large Synoptic Survey Telescope. To address the general problem of analyzing multidimensional datasets (from images, simulations, and others) commonly used in science, our group has partnered with the SciDB project [20]. Our focus has been on the storage and execution layers of this engine. In the context of the AstroDB collaboration, we also built the AscotDB system [23] that enables the efficient analysis of raw telescope images using SciDB. AscotDB extends SciDB with efficient iterative processing features, support for spherical coordinates, and an integrated set of graphical and Python interfaces geared toward telescope image analysis.

3. USER-FACING TOOLS

The number of users who need to manage and analyze large databases is growing much faster than the number of database administrators and developers. Users today thus need to be self-sufficient and modern data management systems must support these users effectively. In response to this trend, our group studies how to facilitate each step of the data management and analysis cycle; we briefly describe here four projects that our group conducted in this space.

SQLShare began as an experiment [8] to try and understand why databases tend to be underused in science — despite a natural fit for question-and-answer hypothesis testing. Common explanations claim a mismatch between scientific data and the models and languages of relational databases, or simply that "scientists won't write SQL." Our hypothesis, instead, was that the core models and languages were more than adequate, but it was too much work to set up a database and get the data in and out. To test the hypothesis we developed a Web-based interface to SQL Azure that reduced the use of databases to a simple upload-query-share workflow that emphasized views and view sharing as the first class interaction mode. The SQLShare experiment has been remarkably successful in demonstrating the utility of databases in new contexts; it currently has hundreds of science users who have uploaded several thousand datasets of varying size and complexity and issued tens of thousands of hand-written SQL queries. We have seen collections of scripts written in R and Python replaced with a handful of SQL queries, simplifying collaborative analysis to the exchange of links into SQLShare [9]. We have seen SQLShare used to facilitate open data and complexity hiding: at least one public dataset is a view that joins 50 distinct tables. In one experiment, we participated in a workshop between 40+ oceanographers from various subfields, using SQLShare to perform "SQL stenography": writing queries in real time to integrate data, test hypotheses, and populate visualizations in response to the live scientific discussion. We found that that ability to write queries in real-time significantly improved the productivity of the meeting, changing it from a "planning" meeting to a science meeting [7].

In a different project, we addressed the core challenge in database usability: assisting inexperienced users in formulating complex SQL queries. This has been a challenge for both research and industry from the early years of relational database systems. Several commercial products include visual query building tools, based on the Query-By-Example visual paradigm, but these are tools aimed at helping users formulate *simple* queries. Our project, SnipSuggest [12], helps users formulate *complex* queries, by taking a radically different approach. As a user types a query, SnipSuggest offers recommendations, by suggesting small *SQL snippets*, such as a list of *k* relevant predicates for the WHERE clause, or a list of UDFs. The key contribution is in selecting relevant recommendations. SnipSuggest produces *context-aware suggestions*: when generating its recommendations, SnipSuggest considers the partial query that the user has typed so far, then draws its recommendations from similar past queries authored by other users, thus leveraging a growing, shared, body of experience. We found this simple idea to have a dramatic impact in practice.

In the context of Cloud service usability, we are currently re-thinking the interface between users and data management services in public Clouds [16]. Today, when a user wants to use a Cloud service, the service asks her to pick a set of resources, such as the number of instances and their specific sizes. When selecting resources, however, users are left wondering: Will the configuration that I pick be fast enough for me? Will I incur any unexpected charges? Will I be able to express the queries that I need (if the service supports one of many existing "SQL-like" languages). To address

Tier 1: $0.10/hour

Within 20 seconds:
SELECT <up to 10 attributes>
FROM <Fact | Dimension>
WHERE <up to 100% of data>

Within 1 minute:
SELECT <up to 5 attributes>
FROM <Fact JOIN up to 4 Dimensions>
WHERE <up to 10% of data>

Within 10 minutes:
SELECT <up to 10 attributes>
FROM <Fact JOIN up to 8 Dimensions>
WHERE <up to 100% of data>

Tier 2: $0.25/hour

Within 5 seconds:
SELECT <up to 10 attributes>
FROM <Fact | Dimension>
WHERE <up to 100% of data>

Within 2 minutes:
SELECT <up to 10 attributes>
FROM <Fact JOIN up to 8 Dimensions>
WHERE <up to 100% of data>

Tier 3: $0.50/hour

Within 1 second:
SELECT <up to 10 attributes>
FROM <Fact | Dimension>
WHERE <up to 100% of data>

Figure 4: Example PSLA with three service tiers.

this mis-match between the resource-centered view of Cloud services and the need-centered view of the users, we are currently developing a new concept of "Personalized Service Level Agreements (PSLAs)" between users and Clouds. Users specify only the schema of their database and the table sizes, while the system generates a PSLA for their database. The PSLA shows a set of price-performance options for queries expressed on the given data. Figure 4 shows an illustrative example of a toy PSLA. We are actively working on this project and incorporating it in Myria.

Finally, in the RFID Ecosystem project [17], we have explored challenges related to helping users manage sensor data. For this, we put together a building-wide RFID infrastructure with 160 antennas covering 7 floors of the CSE building and 67 users carrying over 300 tags. We developed Cascadia, a complete software stack for managing RFID data [24], built a battery of applications, and conducted longitudinal studies. Through these studies, we were the first to characterize the performance of a user-centered, real-life RFID deployment.

4. PROBABILISTIC DATABASES

Probabilistic databases have emerged as a general abstraction for modeling uncertain data, where the presence of records or the values of attributes may be uncertain [21]. When a query is evaluated on such data, each row returned by the query is annotated with a probability. The key research challenge is to compute these probabilities efficiently, hopefully within current query processing engines. This problem has been shown to be closely related to lifted probabilistic inference in Statistical Relational Models, such as in the popular Markov Logic Networks, also developed at the University of Washington [5]. In *exact inference* one asks for the correct output probabilities, while in *approximate inference* one can tolerate errors.

Over the last decade our group has fully described the complexity of exact inference problem in probabilistic databases. First, for some queries, computing the output probability is #P-hard in the size of the database; the simplest example is:

```
select distinct R.A
from R, S, T
where R.B = S.B and S.C = T.C
```

If each record in R and T is an independent random variable, with a given probability of being present, then computing the probability of any output tuple is #P-hard in the size of the relations R, S, T. Other queries, however, can be computed in polynomial time (drop the relation T in the query above and the new query is in PTIME), and, moreover, they can be computed by a query plan where each query operator manipulates the probabilities explicitly (e.g. a join would multiply the probabilities). The main outcome of our research is a complete classification of Unions of Conjunctive Queries (a.k.a. select/project/union/join queries) into two classes, #P-hard or PTIME, forming a dichotomy [4]. A compiler can decide, for any given query, in which of these two classes the query belongs, through a simple static analysis.

Computing the output probabilities is equivalent to weighted model counting on the query lineage[2], on which there exists a rich literature. An alternative approach to pushing probabilistic inference inside a database engine is to first compute its lineage, then use one of the weighted model counting algorithms. We have proven, however, that there exists queries that are computable in polynomial time, yet classic weighted model counters based on the Davis-Putnam-Logemann-Loveland (DPLL) will take exponential time, even with modern extensions such as caching and components. The results of this research are shown in Figure 5. The query evaluation algorithm that resulted from our research is both more powerful than traditional weighted model counters[3], and has the extra benefit that it can be performed within existing query engines.

Our current research focuses on extending the dichotomy to queries with negation, and approximate inference algorithms for #P-hard queries.

5. CAUSES AND EXPLANATIONS

Our group has conducted pioneering research into understanding causality in data transformation, and computing explanations for observed query outputs. Figure 6 shows the number of publications in SIGMOD during a moving five years window, broken down into papers published by authors from industry and papers

[2]In Statistical Relational Models the lineage is called *grounding*.
[3]Our algorithm uses the inclusion/exclusion formula on the query expression, which has no efficient counterpart on the grounded representation (lineage).

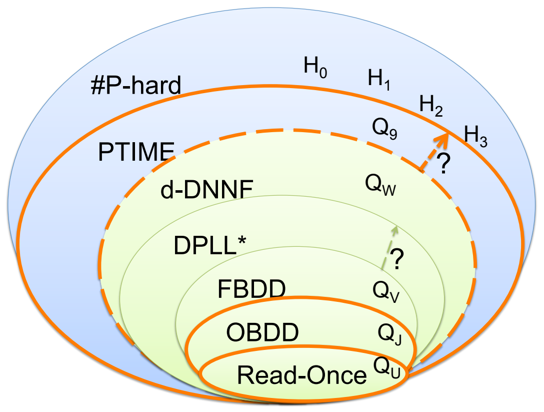


Figure 5: Classification of queries by their complexity on probabilistic databases: some queries are #P-hard, others can be computed in PTIME. Solid lines mean that a complete syntactic characterization of queries in that class is known; dashed line means it is conjectured; thin lines mean it is open. DPLL$*$ means DPLL extended with caching and components.

published by authors from academia. Around 2000-2007 one can notice an interesting bump: while before that period both the number of academic and industrial papers increase over time, after that period the number of academic papers continues to increase, but that of industrial papers decreases. Often users face such unexpected query outcomes and would like explanations to their observed outcome.

The golden standard for an explanation is the *actual cause* of the observed outcome, which has been recently studied and defined algorithmically by Judea Pearl. At the core, causality is defined in terms of intervention: an input is said to be a cause if we can affect the output by changing just the value of that input, while keeping all others unchanged. An *explanation* lowers the bar of causality, and only requires that a change in the input affects the output: the more it affects the output, the better the explanation.

Research in our group ranges from theoretical investigations of the complexity of causality, to query explanation systems. PerfXplain, gives explanations for performance observations in MapReduce systems [11]. PerfXplain logs a number of parameters during the execution of MapReduce programs, then allows the user to ask questions such as *why did my job take the same amount of time even though I have doubled the number of workers?*, and returns ranked answers, such as *because the block size was too large* (and, therefore, there was not enough parallelism available). In a more recent system we have extended explanations to arbitrary SQL queries, such as the one that produces the graph in Fig-

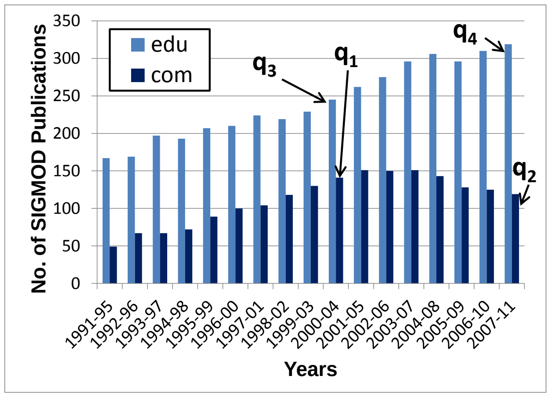


Figure 6: Number of SIGMOD publications in five years windows, broken down into papers from industry ('com') and academia ('edu'). (Affiliation data was available only for some authors, therefore the graph does not include all papers, while papers with authors from both industry and academia are included in both bars.) While both increase until 2000-2007, afterwards the number of papers from academia continues to increase while that from industry decreases. In our research, we study concepts and develop tools that help users find explanations for observed query outputs.

ure 6: the user asks *why is the graph of SIGMOD publications from academia always increasing, while that from industry is increasing much less?* (we refer the reader to [18] for the answers).

6. MANAGING VALUABLE DATA

We are entering an era where the value of technology shifts from the hardware and software artifacts to the data: the most successful companies are those that hold, can produce, or acquire unique and valuable data, such as geographical data (Google maps), social data (Facebook), corporate data (Dun&Bradstreed), maps (Navteq), or miscellaneous data extracted, integrated, and cleaned (Factual). Data has value, and users are keenly aware of that. Some of our most forward-looking research projects develop revolutionary concepts to reason about and manage data with explicit value. We have pioneered the concept of *arbitrage-free query pricing*. Instead of charging per data item, or for an entire dataset, we have proposed a framework where data sellers can charge for every query, based on the information content of its answer. A key property that must be satisfied by such a framework is that the price function be arbitrage-free: if a query Q_1 can be answered entirely from the output of some other query Q_2, then its price should not be larger (otherwise a middleman would purchase Q_2 and resell Q_1 at a profit). This concept should be quite familiar to the database community: it is precisely when Q_1 can be answered entirely from a materialized view Q_2.

In another project, we have looked at the terms of use that accompany data bought and sold on the Web. When major corporations acquire data, they must enforce that their employees follow these terms, and this is difficult: in an informal survey of 13 data providers we found that the average length of terms of use is about 8.3 pages of language full of legal terms, almost impossible to understand and remember by a typical programmer. We are currently developing a system, called DataLawyer, where such policies can be specified declaratively (in SQL) and which can enforce them automatically at query time, by adding only a small overhead to the query processing time [22].

7. EDUCATION

Our group places a strong emphasis on teaching. At the undergraduate level, we teach a course that focuses on using data management systems and building database applications (CSE 344) and another that focuses on building data management systems (CSE 444).

At the graduate level, funded by an NSF IGERT grant, together with colleagues in Computer Science, Astronomy, Oceanography, Genome Sciences, Statistics, and Chemical Engineering, we are putting in place a new PhD program in Data Science [10]. The goal of the program is to create an inter-disciplinary cohort of PhD students who will all specialize in both methods for data science and at least one application domain. Our graduate database course (CSE544) is one of the required courses in this new program.

Through our partnership with the eScience Institute, we have also been engaged with multiple efforts to develop "data science" curricula for both majors and non-majors. One of us developed a Massively Open Online Course (MOOC) called Introduction to Data Science that saw over 9,000 students complete all assignments and over 7,000 earn a certificate. Unlike many courses in this area, we strongly emphasized the central role of database formalisms and technology in data science, pointing out, for example, the ubiquity of the relational algebra even among "NoSQL" technologies. In addition to this online course, we have bootstrapped certificate programs targeting working professionals in cloud computing and data science, we co-developed a new data-oriented introductory programming course for non-majors called Introduction to Data Programming, and we have co-hosted multiple workshops for introductory programming for scientists and engineers through Software Carpentry[4].

[4]`http://software-carpentry.org/`

8. REFERENCES

[1] AstroDB: methods and tools for Big Data astronomy. `http://db.cs.washington.edu/astrodb/`.

[2] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *PODS*, pages 273–284, 2013.

[3] Y. Bu, B. Howe, M. Balazinska, and M. Ernst. The HaLoop approach to large-scale iterative data analysis. *VLDB Journal (Special Issue: Best of VLDB 2010)*, 21(2), 2012.

[4] N. N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30, 2012.

[5] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[6] University of Washington eScience Instittute. `http://www.escience.washington.edu`.

[7] D. Halperin, K. Weitz, B. Howe, F. Ribalet, M. A. Saito, and E. V. Armbrust. Real-time collaborative analysis with (almost) pure SQL: A case study in biogeochemical oceanography. In *SSDBM*, 2013.

[8] B. Howe, G. Cole, E. Souroush, P. Koutris, A. Key, N. Khoussainova, and L. Battle. Database-as-a-service for long tail science. In *SSDBM*, 2011.

[9] B. Howe, F. Ribalet, D. Halperin, S. Chitnis, and E. V. Armbrust. Collaborative science workflows in SQL. *Computing in Science & Engineering, Special Issue on Science Data Management*, 15(2), May/June 2013.

[10] Big Data U: PhD Program in Big Data and Data Science. `http://data.washington.edu/`.

[11] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: Debugging MapReduce job performance. *PVLDB*, 5(7):598–609, 2012.

[12] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. SnipSuggest: Context-aware autocompletion for SQL. *PVLDB*, 4(1):22–33, 2010.

[13] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia. SkewTune: Mitigating skew in MapReduce applications. In *SIGMOD*, 2012.

[14] Myria: Big Data management as a Cloud service. `http://myria.cs.washington.edu/`.

[15] Nuage: Scientific data management in the cloud. `http://nuage.cs.washington.edu/`.

[16] J. Ortiz, V. T. de Almeida, and M. Balazinska. A vision for personalized service level agreements in the cloud. In *Proc. of the DanaC Workshop*, 2013.

[17] The RFID Ecosystem. `http://rfid.cs.washington.edu/`.

[18] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD*, 2014. To appear.

[19] SQL Share. `http://sqlshare.escience.washington.edu`.

[20] UW SciDB Branch. `http://scidb.cs.washington.edu/`.

[21] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[22] P. Upadhyaya, N. R. Anderson, M. Balazinska, B. Howe, R. Kaushik, R. Ramamurthy, and D. Suciu. The power of data use management in action. In *SIGMOD*, pages 1117–1120, 2013.

[23] J. Vanderplas, E. Soroush, S. Krughoff, M. Balazinska, and A. Connolly. Squeezing a big orange into little boxes: The AscotDB system for parallel processing of data on a sphere. *IEEE Data Eng. Bulletin*, 36(4):11–20, 2013.

[24] E. Welbourne, K. Koscher, E. Soroush, M. Balazinska, and G. Borriello. Longitudinal study of a building-wide RFID Ecosystem. In *MOBISYS*, 2009.