Bulletin of the Technical Committee on

Data Engineering

September 2012 Vol. 35 No. 3

IEEE Computer Society

Letters

Special Issue on Data Management beyond Database Systems

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TC on Data Engineering web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

IEEE Computer Society News

I wrote in the last issue about a proposal that TC Chairs had heard from the Computer Society about enabling TC's to maintain a fund balance, and I indicated that we did not have specifics then. The specifics have now been announced, and I do not believe that the current proposal changes very much. The critical point is that this is an extremely limited proposal. Here is the wording: "TCs will have the ability to reinvest surplus allocation for *one additional year* in order to fund new initiatives and expanded activities".

My reply via email was the following: "So- we have two years to consume our allocation. This is truly a very very modest change- and not conducive to long range planning for activities that require some sustained control over budgets. I doubt that this will change behavior of conference planners- who will continue to try to just break even."

TCDE Chair Election

I want to draw your attention to the "Message from the TCDE Nominating Committee Chair" letter from Paul Larson and Masaru Kitsuregawa on page 3. We have two nominees for TCDE Chair, Malu Castelanos and Kyu-Young Whang. Both are distinguished members of the database community who have previously served the TCDE. The TCDE chair position is an important one for the ongoing vitality of the TCDE. I would urge you to think about whom you would like to have in this office and to vote in this election.

Bulletin Editors

The most important part of my job as editor-in-chief of the Bulletin occurs regularly every two years. This is the appointment of new editors. It is these editors who are responsible for the content of the bulletin. It is their efforts that ensure the continued high quality and usefulness of the papers that appear in the bulletin. Appointing quality editors is not only essential for me but also a source of great pride. The people who are willing to take on this job have outstanding research reputations and exploit their knowledge of the field to ensure that each issue of the bulletin becomes a great resource for our community.

Thus, I am very proud to announce that Juliana Freire of the Polytechnic Institute of NYU, Paul Larson of Microsoft Research, Sharad Mehrotra from UC Irvine, and S. Sudarshan from IIT Bombay have agreed to be the next set of editors. Each of them will be responsible for two issues over the coming two years. I want to thank them for agreeing to take on this job, and look forward to working with them.

Simultaneous with the appointment of new editors, the terms of the current editors expire. Over the past two plus years, we have had outstanding Bulletin issue on some of the most important and high visibility areas within the database technology world. These issues were brought to you via the efforts of editors Peter Boncz, Brian Cooper, Mohamed Mokbel, and Wang-Chiew Tan. Once again, I am in awe of the wonderful work done by Bulletin editors. Thank you Peter, Brian, Mohamed, and Wang-Chiew, for your talented and successful efforts over the past two years in keeping the Bulletin such a valuable resource to our community.

The Current Issue

Data management is a term that we use when we do not want to be confined to speaking only about more or less standard database systems. The fact is that data management goes almost ubiquitously. Some of that will very comfortably and effectively rely on database management systems as we currently understand them. But data is incredibly diverse, and so are the applications that use it. So regardless of how you may feel about the "one size does not fit all" characterization of DBMSs, in the broader data management, it is irrefutable that multiple approaches are needed.

With this in mind, Juliana Freire, one of our new editors, has brought to the bulletin a sample of some of the applications that are driving the evolution of data management. Many of these applications are technical in nature, but not all of them. I am truly impressed by the variety of applications and the diversity of their data management needs. The issue topics range from journalism to bio-image data- and these are just the end points in the table of contents. This is not a spectrum, this is a scatter chart of important things happening in a rich and high dimensional application world.

This issue can serve as source material when looking for new data management problems, and can also serve as a way to validate whether a system, perhaps your research system, can deal effectively with requirements from real applications. I want to thank Juliana for producing the issue and can strongly recommend it to Bulletin readers.

David Lomet
Microsoft Corporation

Message from the TCDE Chair Nominating Committee

The Chair of the IEEE Computer Society Technical Committee on Data Engineering (TCDE) is elected for a two-year period. The mandate of the current Chair, David Lomet, is coming to an end and it is time to elect a Chair for the next two years.

The TCDE Chair Nominating Committee, consisting of Masaru Kitsuregawa and Paul Larson has nominated two candidates for the position as Chair of TCDE: Malu Castellanos and Kyu-Young Whang. A call for nominations was published in the June issue of the Data Engineering Bulletin but no other nominations were received.

More information about TCDE can be found at http://tab.computer.org/tcde/.

Paul Larson and Masaru Kitsuregawa
Microsoft Corporation and University of Tokoyo

Election for Chair of IEEE Computer Society TC on Data Engineering

The 2012 TCDE Chair election is now open. The Chair's term will run from January 1, 2013 through December 31, 2014. Before entering the poll, please read the following:

- Please take a few moments to review candidate Biosketches and Position Statements in advance of voting at http://www.computer.org/portal/web/tandc/tcde
- To cast your vote, you will need your IEEE CS Member number (to obtain a misplaced number, please visit http://www.ieee.org/web/aboutus/help/contact/index.html).

Vote here: http://www.surveymonkey.com/s/BQCNFWG

Note: Only CS Members can vote. You may vote only one time.Polls close October 31, 2012, at 5 pm Pacific time.

If you have trouble voting, please contact T&C Sr. Program Specialist, Carrie Walsh (ccwalsh@computer.org).

To ensure that your TC membership is valid and up to date, please log in and update your contact information and TC affiliation at
https://www.ieee.org/membership-catalog/productdetail/showProductDetailPage.html?product=CMYDE709.
This requires an IEEE Web account, which is available free at http://www.ieee.org/go/create_web_account.

Thank you for your participation in this election.

Carrie Walsh
IEEE Computer Society

TCDE Chair Candidate Malu Castelanos

Biography

Malu Castellanos is a senior researcher at HP Labs where she has been working on different areas of data management since 1997; physical database design, real-time business intelligence and text analytics being her primary research areas in the last years. Earlier she was on the Faculty of the Department of Information Systems at the Polytechnic University of Catalunya in Barcelona, where she received her Ph.D, and on the Faculty of the Engineering School at the National Autonomous University of Mexico (UNAM) from where she graduated as computer engineer. She has authored over 60 papers and has 16 patents. She has served in the organization and PC of numerous international conferences and workshops including being General Chair of ICDE 2008 (Cancun) and Organizer of SMDB at ICDE 2012.

Position statement

The TCDEs charter is of high relevance to the data management community as it sponsors ICDE, its flagship conference and one of the top international conferences in this area, and makes decisions on co-sponsoring a limited number of other conferences and workshops. It also promotes the formation of working groups in relevant, emerging and challenging topics to grow strong forums with presence and continuity at ICDE. If elected, I look forward to do my best to continue the excellent work of the current TC chair, David Lomet, in all these activities. I am enthusiastic about preserving the great reputation that ICDE has gained over the years and further increase its visibility and quality for which I will work closely with Masaru Kitsuregawa, chair of the ICDE Steering Committee, and its members. All along, I will seek input and suggestions from the TC members and from David Lomet and his predecessor Paul Larson.

Malu Castelanos
HP Corporation
Palo Alto, CA

TCDE Chair Candidate Kyu-Young Whang

Biography

Kyu-Young Whang is a Distinguished Professor at Computer Science Department of KAIST. Before joining KAIST in 1990, he was a Research Staff Member at IBM T. J. Watson Research Center, New York. His research is mostly in database systems including query processing/optimization, physical database design, indexing, and more recently, in DB-IR integration and search engines. He is an author of over 120 papers in major journals and conferences. He served the research community in various capacities including the coordinating Editor-in-Chief of The VLDB Journal, the general chair of VLDB2006, a PC co-chair of VLDB, ICDE, and CoopIS, a steering committee member of IEEE ICDE, a trustee of VLDB Endowment, and the steering committee chair of DASFAA. He earned his PhD from Stanford University. He received a best paper award and a best demonstration award from IEEE ICDE. He is a Fellow of the IEEE and ACM.

Position statement

The Technical Committee on Data Engineering (TCDE) of The IEEE Computer Society focuses on various aspects of data management and systems. It is an IEEE organization leading and serving the worlds database research community. It sponsors IEEE ICDE, a premier database conference, and co-sponsors a few other conferences and workshops. If elected, I will continue the fine effort of my predecessors, Paul Larson and David Lomet, to promote database research, work closely with the ICDE steering committee to strengthen the flagship conference, and work with VLDB Endowment and ACM SIGMOD for better collaboration.

Kyu-Young Whang
KAIST
Seoul, Korea

Letter from the Special Issue Editor

The explosion in the volume of digital data and its wide availability is revolutionizing many domains. While science is becoming increasingly data intensive, in industry, data has become a commodity — a key means to attain efficiency and generate revenue. This explosion has also democratized data access. Large-scale data analysis and management have historically been carried out in silos with the assistance of highly-trained database experts. Now, masses of data enthusiasts, from scientists to journalists with no knowledge of databases (or the means to pay for experts), are faced with complex analysis tasks and the need to manage data. Even though there are robust and efficient databases management systems, these are hard to configure and use and thus, out-of-reach for data enthusiasts. The problem is compounded due to the fact that complex analyses often involve data that are diverse and require operations that go beyond what is supported by relational databases or warehouses. Consequently, users need to weave complex workflows that combine multiple tools — a task that is difficult even for experts. These new users and applications present new challenges to database research as well as a great opportunity for our community to have practical impact.

In this issue, we have collected a set of articles that highlight new directions for database research that addresses some of these challenges; examples of how database research has been successfully applied to problems that involve large-scale unstructured and structured data in different domains; and articles that relate limitations in current database technology from the point of view of users of the technology. Halevy and McGregor discuss challenges data journalists face and how tools such as Google Fusion Tables, that combine data management and visualization, have given journalists more power to discover and tell stories. Howe and Halperin argue that, in spite of numerous success stories, database systems are underused in the long tail of science, where spreadsheets abound. They describe SQLShare, a web-based query-as-a-service system developed at the University of Washington, that departs from the conventional database design, instead emphasizing a simple Upload-Query-Share workflow and exposing a direct, full-SQL query interface over raw tabular data. They relate real use cases for their system and how it has improved data analysis and management in different scientific domains. Fekete and Silva examine the interplay between databases and visual analytics. They observe that visualization researchers often re-implement database functionality in their tools in a sub-optimal way and discuss a number of data management services and features that are needed for visual analytics. They also posit that research in the intersection of databases and visual analytics would enrich both fields. Li and Jagadish explore the interaction between human computer interaction and databases, and discuss both lessons the database community can learn as well as new research challenges that arise for database usability. Weikum et al. give an overview of scalable techniques for data and text analytics that have been successfully applied in computational linguistics. They leverage big data on the Web and other resources to enhance semantics-centric tasks dealing with natural language texts. Kvilekval et al. discuss the challenges involved in managing biological images and how Bisque, an image database platform they have developed, addresses the limitations of traditional databases and standalone analysis tools for managing and exploring image collections.

There are two recurring themes in these papers. First, the importance of cross-domain synergies. Not only can database research benefit other areas, but we can also learn from and derive new research questions based on the needs of other areas, both within computer science and across different scientific domains. Second, these papers also highlight the need to go beyond traditional database systems and to make database technology more accessible—both easier to use for end-users and easier to integrate with other systems. I hope this special issue will energize discussions around these themes.

I would like to thank all of the authors who agreed to share their work and experiences, as well as Dave Lomet who has provided invaluable guidance during the process of putting this issue together.

Juliana Freire
Polytechnic Institute of New York University
New York, New York

Data Management for Journalism

Alon Halevy
Google Research

Susan McGregor
Columbia School of Journalism

Abstract

We describe the power and potential of data journalism, where news stories are reported and published with data and dynamic visualizations. We discuss the challenges facing data journalism today and how recent data management tools such as Google Fusion Tables have helped in the newsroom. We then describe some of the challenges that need to be addressed in order for data journalism to reach its full potential.

1 Introduction

For decades, computer-assisted reporting (CAR) has been an essential aspect of enterprise and investigative reporting, using compilations of public records, private databases, and other specialized data sources to reveal newsworthy patterns and anomalies, or simply to identify leads for further investigation using more traditional reporting techniques. Yet while personal computers have been a newsroom mainstay for many years, CAR has remained an essentially niche practice in the newsroom, as most reporters have been ill-equipped to exploit the often powerful resources available on their hard drives. The integration of data analysis into mainstream reporting has been stymied by a range of obstacles: arcane and obscure data formats requiring highly specialized languages and skills to manipulate, limited technologies to support visualization and pattern-recognition, data-literacy and numeracy challenges among reporters, and outdated or difficult-to-obtain data sets. Increasingly, news organizations must compete with social media to break news, and the expertise overhead and time lag involved in obtaining, cleaning, and analyzing data has made it impossible to use for deadline reporting. Recent innovations in Web-based data management tools, however, have begun to dismantle some of these obstacles, giving rise to the broader practice of data journalism, which is quickly becoming a core technique of the 21st century newsroom. In this paper we describe how two such tools - Google Fusion Tables [9] and Google Refine [10] - have impacted data-driven reporting, and we describe the next set of challenges to empowering data journalism.

We begin with an example that illustrates the power of data journalism and its growing role in public discourse. On August 11, 2011, British Prime Minister David Cameron addressed an emergency parliamentary session called in response to the widespread UK riots of the preceding week. In his remarks, he stated that *"Everyone watching these horrific actions will be struck by how they were organised via social media,"* and suggested that the UK government was exploring the possibility of limiting or banning access to social media during periods of public unrest [12].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Since the Arab Spring of 2011, the role of social media as an organizing tool for demonstrations and protests had become accepted wisdom in many circles, providing a basis for Cameron's assertion that the government should have the power *"to stop people communicating via these Websites and services when we know they are plotting violence, disorder and criminality,"* [12]. Less than two weeks later, however, a preliminary analysis and visualization of more than 2.5 million tweets by the Guardian UK [4] indicated that riot-related traffic on the Web service tended to spike *after* violence began in a particular neighborhood, as shown in Figure 1. More formal analyses conducted in partnership with the London School of Economics and the University of Manchester over the following months confirmed that messages shared through social media outlets like Twitter and Facebook were not a factor in organizing the riots. In fact, they played a vital role in mobilizing cleanup efforts [1].

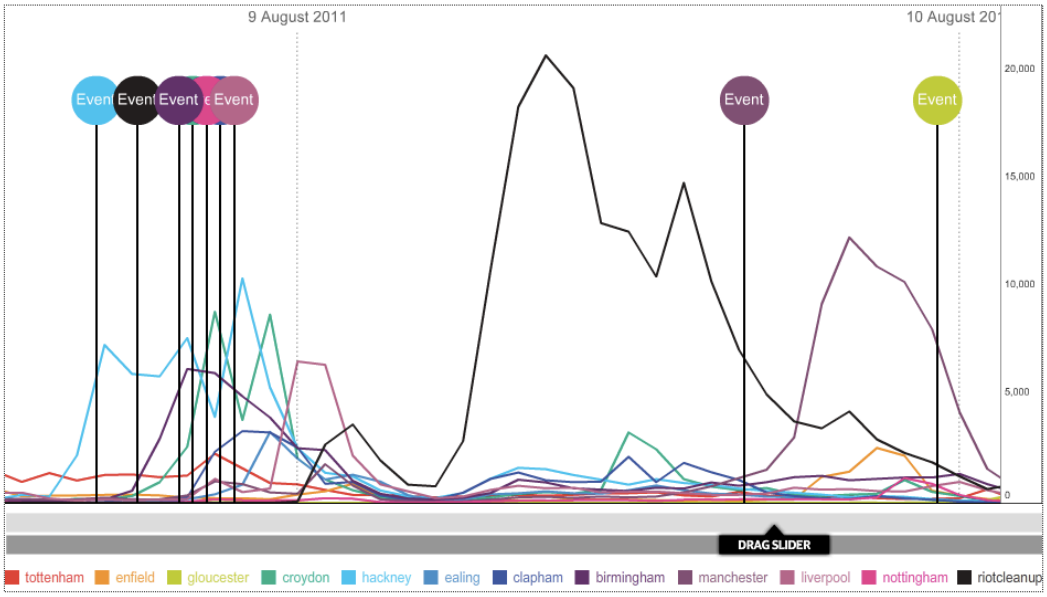


Figure 1: A screenshot of the interactive line chart published by the Guardian on its website on August 24, 2011. The large spike in the middle, shown in black, indicates tweets related to the riot cleanup.

As the Guardian's work demonstrates, there remains an elemental need for professional journalism executed with relevance, rigor and accuracy. Though one can only speculate what actions the government may have taken in the absence of the Guardian's reports, there can be little doubt that the public understanding of the UK riots would have been importantly diminished without them.

This type of insight and public empowerment is at the heart of all great journalism, but is increasingly the purview of data-driven analyses in the newsroom. Whether it's the Guardian's work defending civil liberties in the U.K., the prompting of privacy legislation in the U.S. triggered by the Wall Street Journal's "What They Know" series or the arrests and reform spurred by the L.A. Times' analysis of government payrolls, data journalism leverages the ability to canvass and correlate information with both a scale and detail that would be impossible through traditional reporting. With more than one-quarter of the world's 2.4 billion people creating or consuming content on the Web [24], the volume of information available as digitally-accessible data can only continue to outstrip the knowledge of even expert individuals. In this environment, the practice of data journalism will only grow in its essential role for the industry as a whole.

2 Challenges to Data Journalism

We describe the main challenges faced by data journalists. Many of these challenges are faced by practitioners in other professions as well, but we describe them from the persepctive and expertise of data journalists.

Discovering high-quality data: While many journalists rely on Web searches to locate data sets and other structured information, indexing such information so that it can be discovered through search engines presents significant challenges. Interestingly, humans and computers face two of the same essential difficulties when attempting to locate and identify high-quality information online.

The first challenge is that a great deal of the structured information that is available online resides in the deep Web. Data on the deep Web is typically accessed through HTML forms (e.g., public records and reports, statistical data). Though the data may be well-structured and high-quality, the actual information is sitting in a back-end database and is available as an HTML page only when a user enters a particular query or set of commands. Such content is often described as part of the *invisible Web* because search engines cannot penetrate the forms and crawl the underlying content. Likewise, interface usability issues, specialized query syntax and sui generis data encodings may make the contained data as inaccessible to human users as to their computer counterparts. Though in some cases journalists may be able to call upon system experts (such as librarians or government employees) for assistance, legal and interpersonal barriers can stymie the best efforts at extracting data from the deep Web or similarly-constructed institutional repositories. While there have been some efforts to surface deep-web data [15] and categorize deep-web sites [2], there is still a long way to go to make this data available to journalists.

Even where structured data exists on the so-called surface Web, there remain significant technical challenges to its discovery and use. While more visible to humans, Web pages that contain tables, HTML lists, or have a repeated structure are in fact quite difficult for search engines to index properly and return as well-ranked results for search queries.

The primary difficulty is that it is hard for search engines to determine which pages on the Web contain high-quality data. Less than 1% of the HTML tables on the Web have good data in them, in part because so many of them are used exclusively for formatting purposes [5]. However, recent work [25] has shown that detecting semantic coherence of a column in a table is an effective signal for determining whether a table has high-quality relational content. For example, a table whose rows all contain the names of tree species is semantically coherent and therefore probably contains useful structured data. While such insights are useful, the current reality is that we still know very little about the semantics of structured data on the Web. Because the only schema information we have about tables are column names (at best), inferring the broader context of data is quite difficult.

These inferential challenges often apply to human users of structured data sourced from the Web as well. Limited metadata - whether the information is accessed through a form or page - often makes it difficult to determine the usefulness of a given data set in answering a particular journalistic question. In cases where sufficient metadata can be accessed through another part of the page or site, human users still face the complex task of "scraping" the site to extract the meaningful data from surrounding markup. Tools like ScraperWiki [23] can help support journalists in this task, but rendering the data in a malleable format is still only a first step. Once accessed, there remain the enormously difficult tasks of correcting, editing and making sense of the data - all before its journalistic relevance can even be assessed.

"Dirty" data: Once a data set has been obtained in an appropriate, non-proprietary format, it most often needs to be corrected or *cleaned* before it can be used. Errors, formatting irregularities, missing values, or unknown conventions must be identified, resolved, and systematically addressed before any kind of analysis can occur. For many years, the go-to tool for data cleaning in newsrooms has been Microsoft Excel, at least in part because of its ubiquity as part of the Microsoft Office suite installed on many PCs. Excel provides basic data sorting

and editing features like find and replace, as well as mathematical and statistical functions for calculating sums, averages, and even standard deviations and z-indexes. Advanced features can be used to create basic charts, graphs, macros, and pivot tables for data analysis.

While the spreadsheet interface implemented by Microsoft Excel and other programs is generally accessible even to novice users, effective use of their data-analysis functions still often requires expert instruction and many hours of practice. Moreover assessing the specific size, contents, and parameters of a given data set constitutes a large part of understanding whether it is relevant to a particular journalistic question, and many spreadsheet tools still require that much of this type of analysis be done by inspection.

Google Refine [10], by contrast, has significantly streamlined these fundamental tasks of data analysis. The main interface - while still following a basic spreadsheet layout - immediately displays the loaded data's row count, and updates this figure whenever it is affected by a user action. Text faceting, for example, instantly displays every unique data value present in the selected column. Because the default text-sort in the summary window is alphabetical, minor misspellings or letter-case differences will often appear adjacent to one another, enabling the user to quicky transform them to the same string, a common data cleaning operation. In addition, Google Refine lets users reconcile cell values with entities in Freebase, providing additional context to the data.

Google Refine also provides functions for quickly viewing subsets of even complex data. For every column that is faceted, selecting a value in the summary window filters the data to show only rows matching that value. This process can be repeated across multiple summary windows, and the filtered dataset can be exported at any time without special configuration. Perhaps most importantly, the process is instantly reversible: all filters can be instantly removed and the full dataset restored by simply closing out the summary windows. Finally, Google Refine records the set of transformations performed on the data as scripts. Hence, these transformations can be applied again if the data is reloaded from the source.

The tools of interrogation: Practicing data journalism, at its most fundamental level, means asking journalistic questions of data. Such data interrogation is best achieved through tools that allow data sets to be correlated, queried, manipulated and transformed. Ideally, any such tool will also execute these functions in a fully reproducible, fully recoverable way, so that results can be checked for accuracy, published for transparency, and assumptions can be tested at little or no cost.

Those who are familiar with the principles of databases will quickly recognize that they are by far more appropriate for interrogating data than any spreadsheet program. Indeed, even the faceting feature of Google Refine discussed above essentially implements the simplest type of querying function supported by traditional database systems, and its automatic macro generation constitutes a transferable, stepwise documentation of data manipulation similar to a SQL query. Yet while there is no question that database systems are incredibly powerful, they are notorious for being difficult to use and requiring very skilled and dedicated personnel to use and manage them. Desktop database systems, such as Microsoft Access and SPSS, are costly, and often result in data sets siloed on individual computers. Web-based systems require server space to be purchased, configured, and secured by specialists. Even where these resources are available, traditional database systems ill-suited for relatively transient data management tasks done by people with limited technical or data management experience.

Additionally, most database systems do not have a straightforward method for associating essential metadata with individual data tables or columns. As discussed above, insufficient context around structured data sets are a severe barrier to their effective (re)use. In journalism, the provenance of a data set is paramount. Even where the source is trusted, the recency, measurement units, encodings and other contextual information (e.g., are these employment figures seasonally-adjusted?) that do not comprise the data itself must be evaluated to determine its appropriateness for a given story. Database systems were built with the main goal of supporting high throughput transactions and running complex SQL queries. Though this feature obviates the need to save multiple views of the same data set or email updated files, database tables' lack of contextual information can be profound. An

emailed file has at least some associated source (the sender) and a timestamp (the date of the email). The text of the email itself is likely to contain some contextualizing information. In the absence of cues about origin and context, even technically networked data (e.g., Linked Data [3]) may fail to live up to one of its foremost potentials: recombination and reuse.

Data integration: In today's publishing environment, leveraging the possibilities of networked data has become so important, in fact, that it has helped redefine one of the most significant terms in information technology today: Big Data.

Traditionally Big Data has been characterized solely by its size - specifically, whether it required a supercomputer to process. The most relevant measure of Big Data today, however, is not the size of the file but the extent of the network [7]. Thus, a single YouTube video becomes Big Data by generating 100 million page views, as occurred with the Kony 2012 campaign [14]; likewise, so do the few hundred insurance companies identified by the Sarasota Herald Tribune through analysis of their networks of financial assets and obligations [13]. Because Big Data comprised of networks and relationships embodies phenomena that affect a large number of people, these types of data sets are especially germane to the journalistic enterprise. Exploring these data sets requires powerful tools for integrating data from multiple independent sources, and the main challenges include large-scale entity and schema resolution.

3 New Online Tools

In recent years, a set of new online data management and visualization tools such as Google Fusion Tables [9] and Tableau Public [16] have given journalists more power to discover and tell stories with data. Both of these systems were inspired in some way by ManyEyes [49], an earlier online visualization system. While ManyEyes focused solely on visualization and collaboration around visualization, Fusion Tables and Tableau provide rich data management features. Consequently, users can explore and query their data before the publish a visualization. In what follows we describe Google Fusion Tables and some of its applications within the field (see [8] for a gallery of examples of Fusion Tables in journalism).

Google Fusion Tables is a Web-based tool that combines and extends aspects of spreadsheet, database, graphing and mapping software that supports real-time, responsive, networked data analysis and publishing. Fusion Tables enables easy import of data from CSV and spreadsheet files, and even guesses the data types of each column (which can also be adjusted by the user). On upload, the user is prompted to add meaningful metadata to the table, such as the source name and Web link, as well as a description of the data (with the useful prompt: *"What would you like to remember about this data in a year?"*). Once loaded, individual columns and even cells can be annotated with format and unit information, revision questions, and more.

Fusion Tables uses sharing to help support collaboration and elaboration around data. Imported tables are private by default – only the owner can see the data and make changes. However, the owner can also choose to share the table with collaborators, giving them permission to view, edit, and/or annotate the data. The owner may also choose to make the table public, making it available in both the Fusion Tables search interface, and for indexing by search engines.

Though Google Fusion Tables defaults to a familiar spreadsheet presentation, it supports the selection, projection and aggregation queries usually reserved to databases. Users can also perform simple joins between data sets, as long as they have read permissions to both. Thanks to its sharing model, Fusion Tables makes it possible to integrate data from multiple independent sources. For example, one can join a table containing the coffee production of different countries (coming from the International Coffee Organization) with data about the coffee consumption per capita (coming from Wikipedia).

Google Fusion Tables also provides tools for instant data visualization, an important component of analyzing data for patterns and anomalies. Through the automatic schema-detection, the system tries to identify columns

that can serve as keys for visualization. For example, Fusion Tables will try to recognize columns with locations and columns with time points. When it does recognize such columns, the visualization menu already enables map or time-line views of the data which can be further configured by the user. As such, Google Fusion Tables supports a very rapid flow from data ingestion to visualization in a variety of forms, including bar, line, and pie charts, as well as fully-styled interactive maps. While some of these visualizations can be generated through advanced use of spreadsheet programs, Fusion Tables' mapping feature deserves special consideration.

Much like data journalism and computer-assisted reporting, mapping in newsrooms has long been the purview of a small set of specially-trained reporters, and for many of the same reasons: creating maps required very expensive, complex software and substantial technical skill. Yet in a data set where location is a relevant parameter, the data must be mapped in order to perform any meaningful analysis; there is no universal mathematical construct that can act as a proxy for geography. Google Fusion Tables' location data type, which can interpret many different forms of location information, including country and city names, latitude/longitude coordinates, street addresses and KML, allows a wide range of geographic information to be mapped instantly. For example, see figure 2, an example of data integration where two data sets are shown on the same map.

The map shown in Figure 2 illustrates the power of data integration and mapping. A few days after the earthquake in Japan in March, 2011, the creator of this map combined two data sets that were developed independently: data about earthquakes since 1973 and data about the location of nuclear plants. This visualization helped address the question that was on the minds of billions of people around the world: would an earthquake in their area trigger the kind of disaster that was unfolding in Japan?

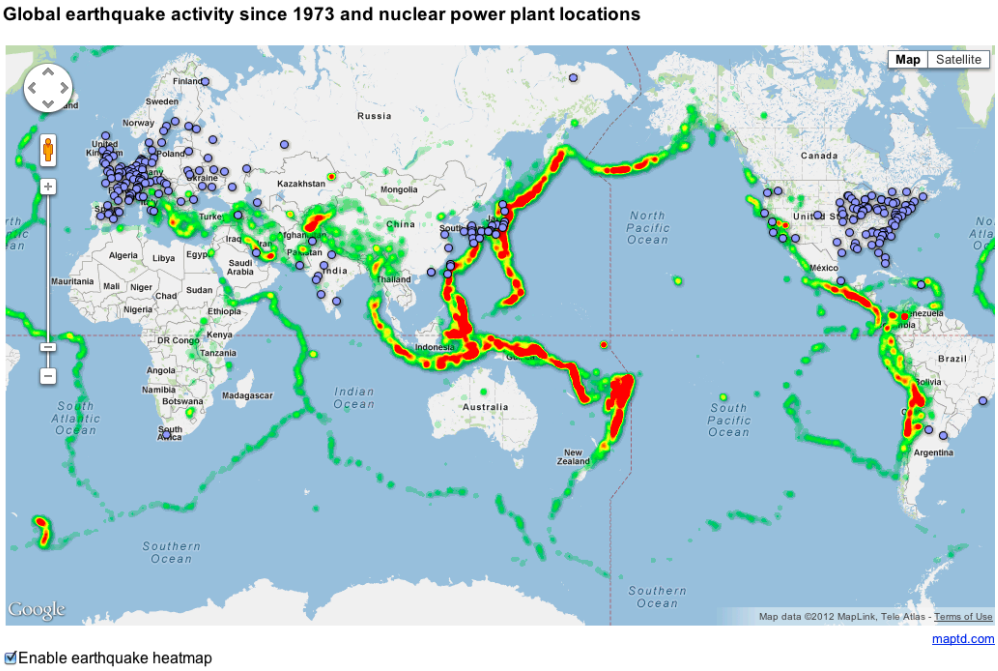


Figure 2: A map that combines two disparate data sources: (1) earthquake activity since 1973, and (2) location of nuclear plants. This map appears on `http://maptd.com`.

4 Remaining Challenges

Thus far, we have touched on some of the tools that are making it possible for more people to ask and answer such questions through the tools and practices of data journalism; below we outline some components needed to realize the full potential of the field.

4.1 Data Literacy

Data literacy needs to be made a priority in all professions, and with it an understanding of the issues around collecting, manipulating, and publishing data. Journalists must cultivate data literacy so that they can evaluate, interrogate and interpret data accurately. Judges must familiarize themselves with the material issues surrounding data formats and accessibility so that they can rule appropriately in data-related cases. Especially in Freedom of Information Act (FOIA) cases, they should require that all metadata and other information necessary to evaluating public data sets be a requisite part of any settlement [17]. As we have reiterated, contextual metadata is the single most defining aspect of a data set's informative value.

4.2 Safety and Privacy

While reporters can now file text, audio, photos – even video – from nearly anywhere, virtually any use of current communication technologies can leave potentially dangerous data traces. While new applications to address these issues are in development [21], both wireless and hard-wired service providers may share user information with government or other entities, endangering sources and journalists. Even if the original requests for user data are later deemed illegal, there may be no legal liability for companies that comply [6]. Used maliciously, this information can create serious threats to civil and human rights.

4.3 Standards and Accessibilty

The sheer number of expensive, complex tools that have been built to work with data is perhaps the clearest indication of its enormous economic value. In the public sphere, however, data should be held to the highest standards of transparency and accountability. Any allegedly public data set should be released in a non-proprietary data format with all metadata intact and relational fields preserved. Unfortunately, this is currently not often the case, as evidenced by the NYPD's release of annual Stop and Frisk data as zipped SPSS archives, and its quarterly reports as pdfs [19].

The journalistic and humanitarian value of standardized data formats is difficult to overstate. For example, many government agencies release weather and other geographic data in KML because it is an accepted global standard. That it is also instantly consumable by Google Maps and Google Fusion Tables means that when there are floods in the midwest, or a tsunami in the Pacific, lifesaving information about what areas are threatened can be published in a matter of minutes [18]. Developing tools and standards that eliminate the need for laborious data cleaning and correlation can improve the speed and accuracy of journalism, increase the transparency and accountability of government, and even save lives.

4.4 The Cloud and the Crowd

Cloud-based data management services such as Google Fusion Tables go a long way to increase the usability of data, in part by making structured data shareable and accessible from anywhere. However, the cloud holds another important promise at the *logical level*. Specifically, if many high quality data sets are put in the cloud, the cloud becomes a rich resource for data that can significantly enhance analysis by enabling data reuse.

Imagine a scenario in which an analyst is looking at the latest trends in health data by county and considering additional locations to focus new efforts. She may be missing critical demographic context in her analysis, such

as the population of each county or its average income - but this type of data is publicly available from reliable sources. If her data is in the cloud, she could potentially just ask for data about population. If the system could examine the locations in her database and find an appropriate dataset that has population data for her locations, her analysis could be dramatically improved.

Adding this contextual data should be so easy that she should not even be aware she performed a database join. Importantly, the provenance of the additional columns should be made very clear, and she should even be able to choose from among any competing data sources. Some of these sources may be branded authorities; others may be crowd-sourced.

While the crowd-sourcing information in this way may seem to contradict our data-provenance requirement, the crowd need not be a nameless set of individuals. Professional communities could collaborate to produce high quality data that they can share through such a system, such as scientists collecting and analyzing data about ecosystems [22], or coffee professionals assembling databases about farms and cafes worldwide [11]. The expertise of these communities might be confirmed by a trusted third party, much like Twitter's "verified user" system; alternatively, user rankings, recommendations or redundant verifications may be used to support sources claims to authority. The project Old Weather [20], for example, successfully crowd-sourced the data entry of naval weather logs by having each scanned record transcribed by multiple users.

With such systems in place, a given community should be able to easily identify coverage gaps and instances where their data quality needs improvement, as well as opportunities to resolve semantic heterogeneity. As these issues are identified and broadcast to the network, the community should be able to go about addressing these issues in a collaborative fashion. Because such a community is comprised of motivated individuals (of many levels of expertise and cost), data collection could be done much more efficiently and effectively than it is today.

5 Conclusions

The latest generation of data management tools has already begun to revolutionize journalism in the 21st century, both in concept and in practice. From the relatively complex, static, and siloed data-manipulation and publishing tools of traditional computer-assisted reporting, the accessible, flexible and networked tools of recent years makes the reach of data journalism virtually as broad as the Web itself. At its core, however, data journalism embodies both the history and the future of the journalistic endeavor. As Pulitzer Prize-winning data journalist Mo Tamman puts it:

"Our job as journalists, more so now than ever, is to help people make sense of what's going on around them. There's so much noise out there it's deafening. Our role is to help them make sense in this deafening noise. It's what we've always done."

References

[1] J. Ball and P. Lewis. Twitter and the riots: how the news spread. *The Guardian*, 2011.

[2] L. Barbosa and J. Freire. Combining classifiers to identify online databases. In *WWW*, pages 431–440, 2007.

[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[4] J. Burn-Murdoch, P. Lewis, J. Ball, C. Oliver, M. Robinson, and G. Blight. Twitter traffic during the riots. http://www.guardian.co.uk/uk/interactive/2011/aug/24/riots-twitter-traffic-interactive, 2011.

[5] M. J. Cafarella, A. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. WebTables: Exploring the Power of Tables on the Web. In *VLDB*, 2008.

[6] Court of Appeals, 9th Circuit 2011 No.09-16676 (Dec. 29)). Hepting v. AT&T. https://www.eff.org/sites/default/files/filenode/20111229_9C_Hepting_Opinion.pdf, 2011.

[7] danah boyd and K. Crawford. Six provocations for big data. In *A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society*, 2011.

[8] Fusion tables gallery. https://sites.google.com/site/fusiontablestalks/, 2012.

[9] H. Gonzalez, A. Halevy, C. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon. Google Fusion Tables: Web-Centered Data Management and Collaboration. In *SIGMOD*, 2010.

[10] Google refine. http://code.google.com/p/google-refine/, 2011.

[11] A. Y. Halevy. *The Infinite Emotions of Coffee*. Macchiatone Communications, LLC, 2011.

[12] J. Halliday. David cameron considers banning suspected rioters from social media. *The Guardian*, 2011.

[13] P. S. John. Insurers risk of ruin. *Sarasota Herald-Tribune*, 2010.

[14] G. Lotan. Kony2012: See how invisible networks helped a campaign capture the worlds attention. http://blog.socialflow.com/post/7120244932/data-viz-kony2012-see-how-invisible-networks-helped-a-campaign-capture-the-worlds-attention, 2012.

[15] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Y. Halevy. Google's deep web crawl. *PVLDB*, 1(2):1241–1252, 2008.

[16] K. Morton, R. Bunker, J. D. Mackinlay, R. Morton, and C. Stolte. Dynamic workload driven data integration in tableau. In *SIGMOD Conference*, pages 807–816, 2012.

[17] No. 10 Civ. 3488 (SAS) (Feb. 7). National day laborer organizing network, et al. v. U.S. Deptartment of Immigration & Customs Enforcement Agency, et al. http://www.ediscoverycaselawupdate.com/National.pdf, 2011.

[18] NOAA. National weather data in kml/kmz formats. http://www.srh.noaa.gov/gis/kml/, 2012.

[19] NYPD. NYPD stop, question and frisk database. http://www.nyc.gov/html/nypd/html/analysis_and_planning/stop_quest 2011.

[20] OldWeather. Old weather: Our weather's past, our climate's future. http://www.oldweather.org/, 2011.

[21] Open internet tools project. http://openitp.org/, 2011.

[22] PCAST Working Group. Sustaining environmental capital: Protecting society and the economy. http://www.whitehouse.gov/administration/eop/ostp/pcast/docsreports, 2011.

[23] Scraperwiki. https://scraperwiki.com, 2010.

[24] I. T. Union. Internet users 06-11. http://www.itu.int/ITU-D/ict/statistics/material/excel/2011/Internet_users_ 01-11.xls, 2011.

[25] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

[26] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. M. McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1121–1128, 2007.

Advancing Declarative Query in the Long Tail of Science

Bill Howe
billhowe@cs.washington.edu

Daniel Halperin
dhalperi@cs.washington.edu

Abstract

Relational databases remain underused in the long tail of science, despite a number of significant success stories and a natural correspondence between scientific inquiry and ad hoc database query. Barriers to adoption have been articulated in the past, but spreadsheets and other file-oriented approaches still dominate. At the University of Washington eScience Institute, we are exploring a new "delivery vector" for selected database features targeting researchers in the long tail: a web-based query-as-a-service system called SQLShare that eschews conventional database design, instead emphasizing a simple Upload-Query-Share workflow and exposing a direct, full-SQL query interface over "raw" tabular data. We augment the basic query interface with services for cleaning and integrating data, recommending and authoring queries, and automatically generating visualizations. We find that even non-programmers are able to create and share SQL views for a variety of tasks, including quality control, integration, basic analysis, and access control. Researchers in oceanography, molecular biology, and ecology report migrating data to our system from spreadsheets, from conventional databases, and from ASCII files. In this paper, we will provide some examples of how the platform has enabled science in other domains, describe our SQLShare system, and propose some emerging research directions in this space for the database community.

1 Introduction

The database community has been incredibly successful at delivering technology to IT professionals, but our tools and techniques remain remarkably underused in science. Despite prominent success stories [8, 24, 2, 11] and what is perhaps a natural correspondence between exploratory hypothesis testing and ad hoc query answering, scientists continue to rely primarily on scripts and files.

The gap is especially acute in the *long tail* of science [19], the small labs and individual researchers who have exploding data requirements but limited IT budgets (Figure 1). For these researchers, data management problems are beginning to dominate their activities: In an informal survey [13], several of our collaborators reported that the ratio of time they spend "handling data" as opposed to "doing science" is approaching 9 to 1!

It may be tempting to ascribe this underuse to a mismatch between scientific data and the models and languages of commercial database systems, but our experience is that standard data models and languages, such as SQL, are suitable for managing and manipulating a significant range of scientific datasets and tasks. We are finding that the key barriers to adoption lie elsewhere:

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

- The initial investment required for the conventional database design and loading process can be prohibitive. Developing a definitive database schema for a project at the frontier of research, where knowledge is undergoing sometimes daily revision, is a challenge even for database experts. Moreover, even a small project may have data in dozens of formats.

- The corpus of data for a given project or lab accretes over time, with many versions and variants of the same information and little explicit documentation about connections between datasets and sensible ways to query them.

- Concerns about control of unpublished research data, at tension with the need for unfettered collaboration and sharing, complicate centralization and organization in a database.

Guided by these premises, we have deployed a new "delivery vector" for SQL targeting researchers called SQLShare [13, 12]. SQLShare de-emphasizes up-front schemas, physical database design, and destructive updates, instead emphasizing logical data independence, ad hoc queries, and controlled sharing. Users upload their data and immediately query it using SQL — no schema design, no reformatting, no DBAs.

Researchers can load data into SQLShare by uploading files through a browser. The system makes an attempt to infer the record structure and schema of the file by recognizing column names, identifying row and column delimiters, and inferring the type of the data in each column. Files with no column headers are supplied with default names. Various data quality issues are addressed automatically: files with an inconsistent number of columns or inconsistent data types among rows can still be uploaded successfully. The design goal was to be as tolerant as possible in getting data into the system and encourage the use of queries and views to repair quality problems. For example:

- Numeric data is often polluted with some string value represeting NULL (e.g., "N/A," "None," or "-"), complicating automatic type inference. The situation is easy to repair by writing a simple view to replace these strings with a true NULL.

- Missing or non-descriptive column names can be replaced by using aliases in the SELECT clause.

- Bad rows and bad columns can be filtered out entirely with an appropriate WHERE clause or SELECT clause, respectively.

- Logical datasets that have been artificially decomposed into multiple files can be reconstructed with a UNION clause. For example, one week of sensor data may be represented as seven one-day files.

This idiom of uploading dirty data and cleaning it declaratively in SQL by writing and saving views has proven extremely effective: it insulates other users from the problems without resulting in multiple versions of the data accumulating, and without requiring external scripts to be written and managed: everything is in the database.

Users create and share views as a first-class activity, one which users have applied to a variety of data tasks: cleaning raw data, integrating disparate sources, and separating public and private data. Some use cases were unexpected: In one case, our collaborators are using deeply nested hierarchies of views as a data processing pipeline, remarking that a stack of views provided several benefits over a sequence of scripts: nothing needs to be re-executed when the pipeline changes, and the view definitions themselves can be inspected directly to provide a form of documentation and provenance. We describe further uses in Section 2.

To help non-experts self-train in writing SQL, we typically provide a "starter kit" of SQL queries when data is first uploaded either manually translated from English questions provided by the researchers as part of our initial engagement protocol (a variant of Jim Gray's "20 questions" methodology [8]), or in some cases derived automatically [14]. These starter queries demonstrate the basic idioms for retrieving and manipulating data, and are saved within SQLShare so they can be copied, modified, and saved anew by the researchers. A cloud-based

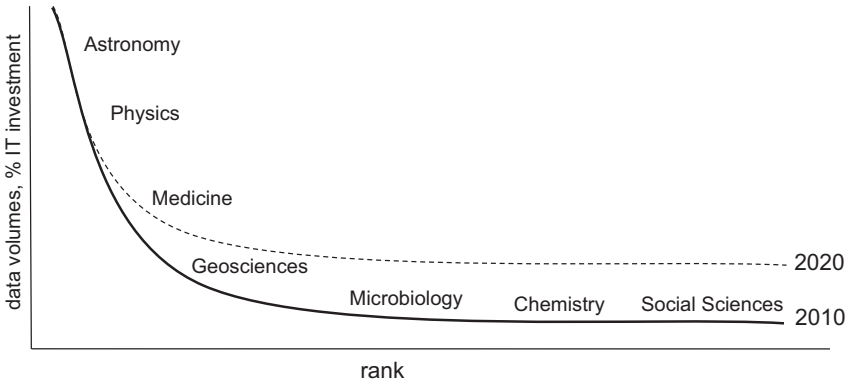


Figure 1: The long tail of science [19] — smaller labs and individual researchers with limited access to IT staff and resources but increasing data management needs and significantly more heterogeneity. These characteristics are more concentrated in some fields. Each line represents a different point in time; data volumes at all scales are trending upward.

deployment on Windows Azure has allowed us to establish a large interdisciplinary corpus of example queries that we can mine to assist in SQL authoring [5], organize and integrate data [3], and, going forward, study the way researchers interact with their data.

The early response from our system has been remarkable. At the first demonstration, the results of a simple SQL query written "live" in less than a minute caused a post doc to exclaim "That took me a week!" meaning that she had spent a week manually cleaning and pre-filtering a handful of spreadsheets and then computing a join between them using copy-and-paste techniques. Within a day, the same post doc had derived and saved several new queries.

The experience was not isolated: the director of her lab has contributed several of her own SQL queries. She has commented that the tool "allows me to do science again," explaining that she felt "locked out" from personal interaction with her data due to technology barriers, relying instead on indirect requests to students and IT staff. She is not alone — over 2000 views have been saved in the SQLShare system by over 200 users since its deployment. In this paper, we describe our initial experience with this system and outline some future directions to explore.

2 Declarative Query for Science: Some Use Cases

It may not be intuitively clear why the SQL language and the extensive use of views is appropriate for scientific inquiry. In this section, we describe a series of examples of how this platform can have an impact on science, especially in the long tail.

2.1 Example: Metagenomics as Set Manipulation

The goal of metagenomics is to correlate environmental conditions with genetic characteristics of the microorganism population as a whole — the metagenome [25]. Intuitively, metagenomics ask two questions: "Who is there?" and "What are they doing?" — i.e., what microorganisms are present in the sample, and what particular metabolic and regulatory activities are active?

Consider a comparative metagenomics experiment involving samples collected at different locations, depths, and times in a river estuary in order to characterize the coastal ecosystem (Figure 3). Each sample may be treated to analyze DNA (identifying organisms) or messenger RNA (identifying expressed genes that have been transcribed), then processed in a "next-gen" massively-parallel sequencing instrument to produce tens of millions of short reads around 35- to 1500-base-pairs long. Population analysis proceeds by using the DNA or RNA reads

to query public reference databases and verifying results against environmental context [17]. Reads without matches may indicate a new species — not unusual, given that hundreds of thousands of distinct species may be typically present in a sample, only a small fraction of which have been described in the literature.

After matching sequences from three distinct samples, one collaborator asked for help translating the following question into SQL: "I need to find the anomalies — genes that are found in at least one of three samples, but not in all of them."

Being a set-oriented declarative language, SQL can express this query rather elegantly: as the union of all samples less the intersection of all samples.

```
(SELECT gene FROM s1 UNION SELECT gene FROM s2 UNION SELECT gene FROM s3)
EXCEPT
(SELECT gene FROM s1 INTERSECT SELECT gene FROM s2 INTERSECT SELECT gene FROM s3)
```

Lessons Learned With our help, this query and others are written and saved within SQLShare, reusable by other researchers as examples. This process — to take users' data and English queries to bootstrap a solution — was described by Gray et al. in the context of the Sloan Digital Sky Survey [9]. Having captured thousands of queries in our system, most of which are written by the scientists themselves after some initial hand-holding, we corroborate SDSS findings that SQL is not the bottleneck to the uptake of relational technology. But we extend these findings to long tail science situations with limited access to dedicated database experts and limited IT funding.

2.2 Example: Integrating Ecological Field Measurements

Microorganisms sustain the biogeochemical cycling of nitrogen, one of the most important nutrient cycles on earth. A key step in this cycle, the oxidation of ammonia to nitrite by autotrophic microorganisms, is now attributed to the ammonia-oxidizing archaea (AOA), which are of high abundance in both marine and terrestrial environments. Understanding the environmental conditions in which these microorganisms thrive may have a significant impact on our ability to manage biodiversity [18].

To study the hypotheses related to how these organisms regulate and control the forms of nitrogen available to other microbial assemblages, field measurements are taken at different times and locations and combined with laboratory assays. Data collection is performed with spreadsheets for maximum flexibility. The raw field measurements are quality-controlled and integrated into "master" spreadsheets. Here the choice of technology breaks down: Shared files and frequent revisions to quality control procedures leads to multiple conflicting versions and poor provenance, processes for transforming data must be repeated by hand, the physical organization of the data into worksheets is inflexible and not always conducive to question answering.

The lab recognized the value in a database to organize these data, but did not want to sacrifice the flexibility of using spreadsheets for data collection. Out in the field, a fixed data entry form is too restrictive. Further, the (obvious) schema was fully expected to evolve steadily over time as they added more measurements.

Lessons Learned SQLShare became attractive as a "staging area" for the data as it came in from the field. With a few queries saved as views, the data could be integrated (the union of datasets from different researchers), cleaned (units standardized and incomplete records suppressed), and shared. Moreover, these views delivered a form of automatic provenance: the hierarchy of composed views, incrementally filtering, integrating, and cleaning the data, are visible to all and refresh automatically whenever the result is accessed. There is no need to "re-run" the data processing pipeline or workflow whenever the data changes. Finally, the use of SQLShare did not preclude the development of a more conventional database application down the road. In fact, a "rough cut" integration effort of the kind facilitated by SQLShare is a necessary first step.

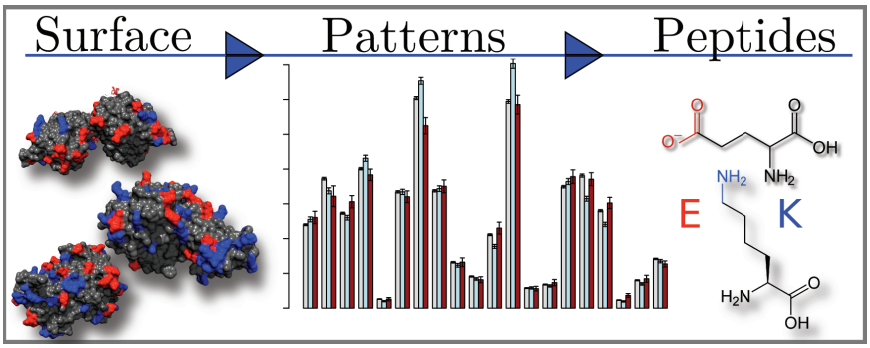


Figure 2: Illustration of the steps in an algorithm to (a) identify the surface of proteins, (b) calculate various statistics, and (c) synthesize "stealth" molecules that could mimic the protein surfaces. The use of SQLShare provided opportunities to exchange "a 10 minute 100 line script for 1 line of SQL."

2.3 Example: Drug Design and Stealth Proteins

A graduate student in Chemical Engineering was working with gigabytes of tabular data derived from analyses of protein surfaces. Given a set of proteins, a series of Python scripts identified their surface (Figure 2(a)), calculated statistics on them (Figure 2(b)), and synthesized molecules that could mimic the protein surfaces (Figure 2(c)). The application area is tissue engineering and implantable materials. The work involves creating surfaces that are "stealth," or invisible to the body because they look like proteins [20].

Loading the data into SQLShare allowed the student and an undergraduate assistant to access the data via the web, download query results for further analysis in R, and — crucially — "give reviewers access to data in publications." Previously, they report using "huge directory trees and plain text files." The adoption of SQL for basic processing allowed them to "accomplish a 10 minute 100 line script in 1 line of SQL [that ran in significantly less time]."

The previous implementation was considerably more complex:

> I ran a Python script that iterated through directories, each of which contained a protein. The Python script called shell scripts which called R scripts to generate data on each protein which was all copied into a total data single directory. Then the Python script calls another shell script which calls R scripts to generate statistics on the total data followed by another set of R scripts to generate plots. What was really getting out of control was that I ended up generating 50 files for each protein, which meant I had 125,000 files in a single directory in the course of the analysis.

Lessons Learned By implementing parts of this pipeline in SQL, the student was able to extract the table manipulation steps out of R and focus instead on the statistics. This separation of concerns is a key goal for SQLShare: it is designed not to supplant general purpose languages, but allow SQL to be as easy to use for quick scripting of table manipulation as R, Python, or MATLAB are for their own strengths. As a side effect, you will often realize significant performance gains even without physical database design and tuning by avoiding "quick and dirty" algorithms that lead to combinatorial explosions of file operations.

3 SQLShare System Details

SQLShare has three components [12]: a web-based UI, a REST web service, and a database backend. The UI is a Django Python application (a web framework similar to Ruby on Rails), and hosted on Amazon Web Services. The UI communicates with the backend exclusively through REST calls, ensuring that all client tools have full access to all features. The web service is implemented on Microsoft Azure as (one or more) Web Roles. The

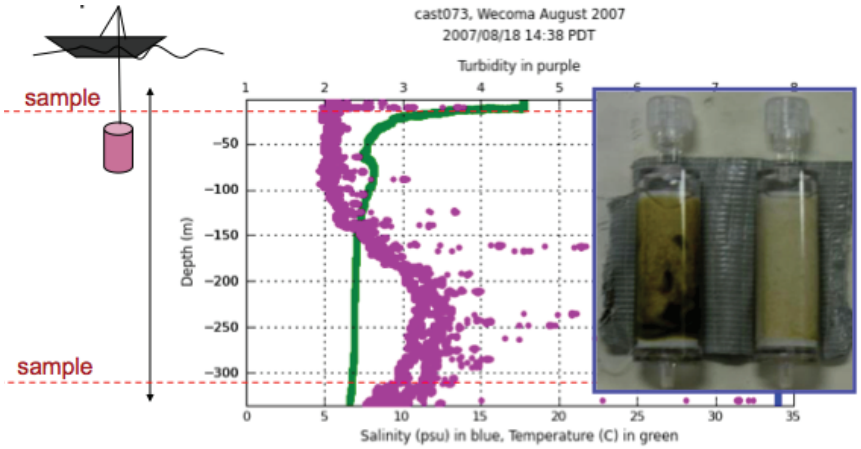


Figure 3: Oceanographic metagenomics involves sequencing entire microbial populations under different environmental conditions. Here, two samples are collected at different depths in the water column from a single *cast* of a Conductivity-Temperature-Depth sensor package (CTD). The sample near the surface has significantly more particulate matter, as is visible once the water is passed through a 2-micron filter (at right). These samples are frozen at sea, sequenced on shore, and computationally compared to correlate environmental conditions with population characteristics. This area of research may involve significantly more data and more samples than genomics techniques involving a single organism.

database is implemented using Microsoft's SQL Azure system, which is very similar to Microsoft's SQL Server platform.[1]

All permissions handling is pushed down into the database. Each SQLShare user is associated with a database user and a schema, and permissions changes in the UI are translated into GRANT and REVOKE statements in the database. Web authentication is handled through OAuth and Shibboleth; once authentication is confirmed, the service impersonates the user when issuing queries.

The SQLShare data model, API, and supported features are designed to lift certain database features (e.g., views) and suppress others (e.g., DDL and transactions). Here is a summary of the distinguishing features:

No Schema We do not allow *CREATE TABLE* statements; tables are created directly from the columns and types inferred in (or extracted from) uploaded files. Just as a user may place any file on a filesystem, we intend for users to put any table into the SQLShare "tablesystem," not just those that comply with a pre-defined schema.

Unifying Views and Tables Our data model consists of a single entity: the *dataset*. Both logical views and physical tables are presented to the user as datasets. By erasing the distinction, we reserve the ability to choose when views should be materialized for performance reasons. Since there are no destructive updates, we can cache view results as aggressively as space will allow. When a view definition changes, downstream dependent views may no longer be valid. In this case, we create a pre-change snapshot of any views invalidated by the change. With this approach, no view will ever generate errors. However, these semantics may not be what the user expects; we are exploring alternatives for communicating these semantics to the user and allowing alternatives in some situations.

Incremental Upload Datasets can be uploaded in chunks. This mechanism allows large files to be uploaded safely, but also affords support for appends: A chunk for a table can arrive at any time, and the table can be freely queried between chunks (the chunked upload is non-transactional.)

Tolerance for Structural Inconsistency Files with missing column headers, columns with non-homogeneous types, and rows with irregular numbers of columns are all tolerated. We find that data need not be pre-cleaned for some tasks (*e.g.,* counting records), and that SQL is an adequate language for many data cleaning tasks.

[1]The differences include: tables must have clustered indexes, non-SQL user-defined functions are not supported, and most distributed query features are not supported.

Metadata and Tagging SQLShare encourages creating views liberally. Navigating and browsing hundreds of views has emerged as a challenge not typically encountered in database applications. To help solve the problem, views can be named, described, and tagged through the UI and programmatically through the REST web service. The tags can be used to organize views into virtual folders. In future work, we are implementing bulk operations on virtual folders: download, delete, tag, change permissions. We are also experimenting with a feature that would allow regex find-and-replace over a set of view definitions to simplify refactoring. We envision eventually evolving into a database-backed IDE-type environment for SQL and UDF development.

Append-Only, Copy-on-Write We do not allow destructive updates. Users insert new information by uploading new datasets. These datasets can be appended to existing datasets if the schemas match. Name conflicts are handled by versioning — the conflicting dataset is renamed, and views that depend on the old version are upated to reflect the change.

Simplified Views We find views to be underused in practice. We hypothesize that the solution may be as simple as avoiding the awkward CREATE VIEW syntax. In SQLShare, view creation is a side effect of querying — the current results can be saved by simply typing a name. This simple UI adjustment appears to be effective — over 2000 views have been registered in the system by over 200 users.

Provenance Browsing We find that some users create deep hierarchies of rather simple, incremental views. This usage pattern is encouraged — the optimizer does not penalize you at runtime, and a composition of simple queries is easier to read and understand than one huge query. However, databases provide no natural way to browse and inspect a hierarchy of views. The catalog must be queried manually. In SQLShare, we are actively developing two features to support this use case: First, a *provenance browser* that creates an interactive visualization of the dependency graph of a hierarchy of composed views to afford navigation, reasoning, and debugging. Each node in the graph can be clicked to access the view definition in the existing SQLShare interface. Second, each table name in a view definition is rendered as a link if it refers to a view, affording more direct navigation through the hierarchy.

Semi-automatic Visualization An immediate requirement among frequent users of SQLShare is visualization. VizDeck is a web-based visualization client for SQLShare that uses a card game metaphor to assist users in creating interactive visual dashboard applications in just a few seconds without training [16]. VizDeck generates a "hand" of ranked visualizations and UI widgets, and the user plays these "cards" into a dashboard template, where they are automatically synchronized into a coherent web application that can be saved and shared with other users. By manipulating the hand dealt — playing one's "good" cards and discarding unwanted cards — the system learns statistically which visualizations are appropriate for a given dataset, improving the quality of the hand dealt for future users.

Automatic Starter Queries SQLShare users frequently do not have significant SQL expertise, but are fully capable of modifying example queries to suit their purpose [23]. For some collaborators, we seed the system with these "starter queries" by asking them to provide English questions that we translate (when possible) into SQL. But this manual approach does not scale, so we have explored automatically synthesizing good example queries from the structural and statistical features of the data [14]. Users upload data, and example queries that involve reasonable joins, selections, unions, and group bys are generated automatically. We are in the process of deploying this feature in the production system.

4 Opportunities and Future Directions: Optimizing for Attention

SQLShare fits into an over-arching theme in eScience: database systems for scientists must be optimized for human attention. Consider Figure 4 (obtained from [1]): Both data volumes and computing capacity are growing exponentially over time, but human cognition has remained essentially flat. For the pace of science to keep up with the rate at which data is being generated, this computational power should be exploited to ensure that scientist attention is utilized to maximum effect.

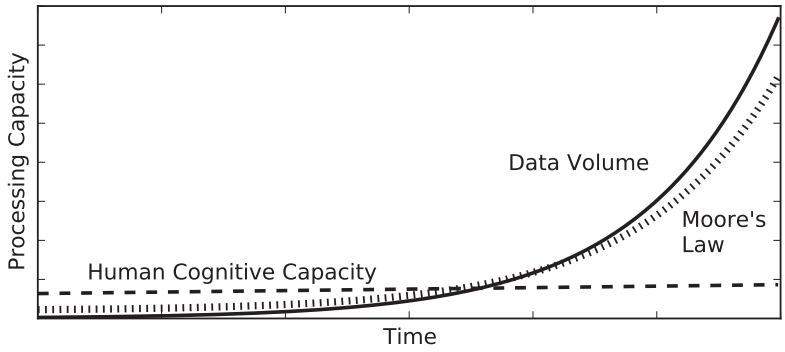


Figure 4: Data volumes grow exponentially, and computing resources have arguably kept pace. However, human cognition has remained essentially flat. This gap represents the dominant challenge for data-intensive science in the long tail.

One direction is to diminish the need for human intervention. SQLShare enables scientists to focus on asking science questions instead of on manual data processing/management or on database administration tasks. We are exploring new techniques that pursue aggressive automation of "human" tasks: SQL authoring [14, 5], visualization [16], and eventually statistical tests.

A second inefficient use of scientist attention is waiting for results. We make two observations about data use in the long tail that may help improve efficiency. First, data access is sparse and infrequent relative to the rate at which data volumes and computational capacity are growing. As a result, even shared data systems for science are typically pathologically underutilized [22]. Second, many science datasets are essentially read-only. Datasets registered as part of the scientific record are never intended to be updated (e.g., historical archives or data associated with a publication). Even prior to publication, data may be grouped into logical batches, then processed and quality-controlled as a unit before appending to an existing dataset (e.g., oceanographic research cruises [4]). These two observations suggest that there are opportunities to aggressively, speculatively, and perhaps wastefully consume resources for even small potential returns in runtime performance. We are exploring a variety of techniques to allow SQLShare and related systems to consume all available resources to improve performance and reduce human effort:

Eager Materialization All query results are cached and retained as materialized views that can be used to optimize future query plans. Even without adopting more sophisticated techniques to answer queries using materialized views [10], identical queries are often run multiple times and can benefit from this cache. In a typical scientific database, the lifetime of intermediate results is long and so is the expected utility of the cached results. This idea is related to database cracking [15], but simpler: Rather than physically reorganizing the data, we propose to simply keep multiple copies of the data. Several challenges emerge: Now that the entire available disk is potentially a cache, eviction policies based on current and anticipated query workload become interesting. These eviction policies can consider the time to recreate the view, partial materialization, and user-specified priorities as well. In addition, a practical implementation of techniques to answer queries using views is worthwhile.

Semantic Prefetching Caching relies on a "warm-up" period to fill the cache. Users receive no benefit querying "fresh" data, and at the frontier of research, most of the interesting data is fresh. Worse, the system will typically sit idle waiting for the human operator to express the next query. As data volumes and processing capabilities grow, the opportunity cost of this idle time becomes increasingly wasteful. We propose a second set of techniques, that we collectively refer to as *semantic prefetching* to exploit this idle time by anticipating unasked queries and speculatively generating possibly-useful results. We envision four levels of semantic prefetching:

1. In a system with geographically-dispersed, prefetch data that is known to already exist based on a model of

the user's task. This capability includes ranking results by similarity based on the user's recent browsing history ("more like this"), by results of interest to one's social network, by global popularity, and by compatibility with explicit user preferences. These techniques are already being explored in a variety of contexts.

2. Given an existing query $q(R)$ for some dataset R, apply q to some new dataset R' that has a compatible schema. For a typical database application, situations where this technique would be useful are essentially nonexistent. But the usage patterns of SQLShare suggest thousands of tables that are closely related (for example, multiple files telemetered from the same sensor, or different versions of the same data under different quality control assumptions). In these cases, users specifically request an operation of the form "do what I just did, but on this table." These opportunities are not difficult to recognize and proactively compute.

3. Even if we do not have an existing query to work from, we may be able to deduce some likely operations: candidates for joins, candidates for group bys, etc. We have begun to explore this approach in our work on automatic starter queries [14].

4. Perhaps neither the data nor the query is currently available in the system. In this case, it seems that all we can do is sit idle. However, if data acquisition itself is under the influence of the database system, as it is, for example, in the context of crowdsourced databases (c.f. Franklin et al. [7]), then there may be opportunities to speculatively acquire data based on predictions of the user's interests.

This last level warrants examples: Under what circumstances might the database be equipped to influence the acquisition of science data?

- Observational oceanographers sometimes make use of the concept of *vessels of opportunity* — vessels that are not under direction of the chief scientist, but that are in the right place at the right time to take an important measurement or sample. Empowering the system to automatically identify these opportunities and issue the request could significantly increase the value of data collected. Currently, scientists may not identify these situations until it is too late to exploit them.

- Citizen science projects provide volunteers with enough training and equipment to collect data on behalf of a research project. For example, volunteers for the Nature Mapping project [6] record observations of wildlife species in populated areas, dramatically improving the quality of the range maps used to inform public policy. A system that could proactively identify regions and species for which little data exists and issue standing requests for additional observations could amplify the effectiveness of these projects, and make the experience more rewarding for the volunteers.

- The term *adaptive sampling* refers to the capability of some sensors to receive commands while operating autonomously in the field. For example, an autonomous underwater vehicle (AUV) may adapt its trajectory based on commands issued from shore, or an atmospheric radar may rotate its antenna to face an incoming storm [21]. A system that can direct these resources automatically based not only on current observations but also based on the value of the potential derived products could increase the return on investment of deployment.

For all four levels, the search space is enormous and cannot be searched directly. Instead, we need to identify promising results by modeling importance to the user. While quantifying importance is difficult, a simplifying factor is that it is not necessarily important that the model is accurate when first deployed, as long as it is equipped to incorporate human feedback and learn statistically what is important and what is not. We envision an interface where scientists can browse the data products derived the previous night over their morning coffee, selecting a few for further review while rejecting the majority. This interaction provides a strong signal on which

to base a ranking algorithm. In some systems, storage resources may be limited. For these cases, cached and prefetched data can be pruned based on expected utility.

5 Summary

We have presented SQLShare, new platform for lightweight, aggressively automated data management designed to 1) allow SQL to compete with other lightweight languages and tools favored by scientists (Excel, Python, MATLAB, ASCII files, R), 2) increase collaborative data sharing by making it easy and fruitful to upload into a web-based system, 3) signficantly reduce the 9-to-1 ratio of time researchers report spending on data handling relative to science. Our progress so far is promising, with even non-programmers responding positively and becoming active users. In addition to the benefits to scientists and helping them explore their data, the use of a simplified interface for querying relational data not only motivate new research problems for the database community, but directly provide a corpus of real data and real queries that can catalyze the study of those problems.

6 Acknowledgments

This research was sponsored by NSF award 1064505, the Gordon and Betty Moore Foundation, and Microsoft Research.

References

[1] C. Aragon. Scientist-computer interfaces for data-intensive science. Microsoft eScience Workshop, 2010.

[2] S. Baker, J. Berger, P. Brady, K. Borne, S. Glotzer, R. Hanisch, D. Johnson, A. Karr, D. Keyes, B. Pate, and H. Prosper. Data-enabled science in the mathematical and physical sciences. Technical report, National Science Foundation, March 2010.

[3] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1), 2009.

[4] Center for Coastal Margin Observation and Prediction. `http://www.stccmop.org`.

[5] Collaborative query management. `http://db.cs.washington.edu/cqms`.

[6] K. Dvornich. Query NatureMapping data using SQLShare. `http://naturemappingfoundation.org/natmap/sqlshare/NM_sqlshare_1.html`.

[7] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: Answering queries with crowdsourcing. In *Proc. of the SIGMOD Conf.*, pages 61–72, 2011.

[8] J. Gray, D. T. Liu, M. A. Nieto-Santisteban, A. S. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *CoRR*, abs/cs/0502008, 2005.

[9] J. Gray and A. S. Szalay. Where the rubber meets the sky: Bridging the gap between databases and science. *CoRR*, abs/cs/0502011, 2005.

[10] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, Dec. 2001.

[11] T. Hey, S. Tansley, and K. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[12] B. Howe. SQLShare: Database-as-a-service for long tail science. http://escience.washington.edu/sqlshare.

[13] B. Howe and G. Cole. SQL Is Dead; Long Live SQL: Lightweight Query Services for Ad Hoc Research Data. In *4th Microsoft eScience Workshop*, 2010.

[14] B. Howe, G. Cole, E. Soroush, P. Koutris, A. Key, N. Khoussainova, and L. Battle. Database-as-a-service for long tail science. In *SSDBM '11: Proceedings of the 23rd Scientific and Statistical Database Management Conference*, 2011.

[15] S. Idreos, M. L. Kersten, and S. Manegold. Database cracking. In *Proc. of the Third Biennial Conf. on Innovative Data Systems Research (CIDR)*, 2007.

[16] A. Key, B. Howe, D. Perry, and C. Aragon. VizDeck: self-organizing dashboards for visual analytics. In *Proc. of the SIGMOD Conf.*, pages 681–684, 2012.

[17] R. Kodner, F. A. Matsen, N. Hoffman, C. Berthiaume, and E. V. Armbrust. A fast, phylogenetic based pipeline for shotgun environmental sequences analysis. *In prep*.

[18] S. N. Merbt, D. A. Stahl, E. Casamayor, E. Marta, G. Nicol, , and J. Prosser. Differential photoinhibition of bacterial and archaeal ammonia oxidation. *FEMS Microbiology Letters*, 327:41, 2012.

[19] P. Murray-Rust and J. Downing. Big science and long-tail science. `http://blogs.ch.cam.ac.uk/pmr/2008/01/29/big-science-and-long-tail-science/`, term attributed to Jim Downing.

[20] A. K. Nowinski, F. Sun, A. White, A. Keefe, and S. Jiang. Sequence, structure, and function of peptide self-assembled monolayers. *Journal of the American Chemical Society*, 134(13):6000–6005, 2012.

[21] B. Plale, D. Gannon, J. Brotzge, K. Droegemeier, J. Kurose, D. McLaughlin, R. Wilhelmson, S. Graves, M. Ramamurthy, R. D. Clark, S. Yalda, D. A. Reed, E. Joseph, and V. Chandrasekar. CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting. *Computer*, 39(11):56–64, 2006.

[22] K. Ren, Y. Kwon, M. Balazinska, and B. Howe. Hadoop's adolescence: A comparative workload analysis from three research clusters. Technical Report UW-CSE-12-06-01, University of Washington, 2012.

[23] Sloan Digital Sky Survey. `http://cas.sdss.org`.

[24] A. Szalay and J. Gray. 2020 computing: Science in an exponential world. *Nature*, 440(7083):413, 2006.

[25] S. Tringe, C. von Mering, A. Kobayashi, A. Salamov, K. Chen, H. Chang, M. Podar, J. Short, E. Mathur, J. Detter, P. Bork, P. Hugenholtz, and E. Rubin. Comparative metagenomics of microbial communities. *Science*, 308(5721):554–7, 2005 Apr 22.

Managing Data for Visual Analytics: Opportunities and Challenges

Jean-Daniel Fekete
INRIA Saclay

Claudio Silva
NYU Poly and NYU CUSP

Abstract

The domain of Visual Analytics has emerged with a charter to support interactive exploration and analysis of large volumes of (often dynamic) data. A common feature shared by all the visual analytics applications developed so far is the reliance on ad-hoc and custom-built mechanisms to manage data: they re-implement their own in-memory databases to support real-time display and interactive feedback and analytical algorithms (e. g., clustering, multidimensional projections, specific data analyses) to overcome the delay required to exchange data with specialized analytical environments, such as Matlab, R, and the myriad of more specialized systems and command-line programs. This article presents visual analytics scenarios requiring a new generation of databases to support and foster the new era of interactive analytical environments. This new generation would relieve visualization researchers from sub-optimally re-implementing database technologies. We argue that the new services required to support interactive explorations present research challenges to the database community and can increase the utility and simplicity of integration of the new generation of databases for data-intensive applications.

1 A Fresh Look at Large Data Visualization

In database research, the big data issue has mainly been addressed as a scale issue: storing and providing the same level of services as before in terms of speed, reliability, interoperability and distribution. The scale challenges are now being solved by research and industry. A less advertised issue raised by the increase of available data is the unprecedented opportunity for discoveries and exploratory studies. For example, Metagenomics is a research field in Biology that sequences DNA from random samples of material collected in the wild to discover the largest possible diversity of living species. With new high-throughput sequencing technologies, the genome of millions of species are sequenced and stored in databases but are not systematically explored due to the lack of appropriate tools. Bank transactions are being logged and monitored to find patterns of frauds, but new schemes arise continually that need to be discovered in the billions of transactions logged daily. This task of discovery of innovative fraud schemes is not well supported by existing tools either. Similar problems arise in a wide range of domains where data is available but requires specific exploration tools, including sentiment analysis on the Web, quality control in Wikipedia, and epidemiology to name a few.

However, current database technologies do not meet the requirements needed to interactively explore massive or even reasonably-sized data sets. In this article, we discuss what the requirements are, why they are not

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

met and, in some cases, what could be done to better support them. Interactive exploration of large data is the goal of the new multidisciplinary field of Visual Analytics (VA), which brings together *visualization*, *interactive data analysis* and *data management*.

Visualization uses the human vision system to understand and find insights on data by displaying it with an adequate visual representation. The visual capabilities of humans far exceed their symbolic capabilities for processing a large number of items: exploring data tables containing thousands of items with tens of dimensions (e. g., all the available models of cameras on the market today with their main features) is a routine task for a modern information visualization (infovis) application but would be very demanding by merely looking at the same data with a standard spreadsheet. However, infovis is limited to the human visual perception that work well with thousands of data items and can, in some specific cases, explore data sets with about 1 million items, but certainly not above. Existing data sets easily exceed this number and still need to be explored. This is why VA combine analytical algorithms in the exploration process to overcome the dimensionality problems. Analytical algorithms can reduce the number of items (e. g., by sampling or aggregating), can reduce the number of dimensions (e. g., by projecting data, or finding dependencies between dimensions), can extract more meaningful items (e. g., by feature extraction) and can help find trends, patterns and outliers. One important challenge when connecting analytical algorithms with interactive visualization is to keep the system interactive and reactive. Most analytical algorithms available off the shelf are meant to process data from start to end regardless of the time it takes. Humans conducting interactive tasks expect results to appear quickly, even if incomplete initially or only rough estimates. Turning the existing analytical algorithms into reactive any-time algorithms is one important challenge that visual analytics needs to address.

In addition to the intrinsic running time of these algorithms, data exchange is currently a bottleneck. Most analytical environments—whether generic such as R or Matlab, or more specific—read the input data from files or databases, perform their computations and write their results in output files or databases. For large data sets, the input/output time exceeds by far "interactive time", the time required to perceive a program as reactive or interactive. To be effective, interactive exploration implies system requirements that are unusual for data management and are therefore not provided in standard databases, forcing VA developers to re-implement their data storage services in an ad-hoc way, usually sub-optimal for the standard services, and hampering interoperability.

We will discuss this issue in more detail in the next section, where we describe systems designed to support visual data exploration and discuss their requirements and limitations regarding data management. We also discuss real applications (UV-CDAT and Wikipedia) that illustrate needs and limitations of standard data management systems. The last section summarizes the opportunities and challenges in light of the requirements outlined before.

2 Visualization Systems

To turn data into visual depictions and provide interactions to navigate, analyze and explore them, visualization needs software libraries that load data, apply transforms through a pipeline of operations to eventually draw the results on a screen and manage input devices to interpret interactions. The two main sub-fields of visualization have slightly different architectures that are described below, with data management issues highlighted.

2.1 Scientific Visualization

The most common data types in scientific visualization systems come from a relatively small set, and can be described succinctly. The data has a given dimension (mostly 1D, 2D and 3D) and data sets are composed of very large collections of cells. The geometry of a cell is determined by its corresponding nodes, and the way the cells are glued together determine their topological structure. Besides geometric properties, there is often associated data (e. g., temperature, wind velocities). Explicitly storing nodes, cells, and their relationships

can be wasteful, in particular since much data belong to well-defined categories that can be stored much more efficiently. Scientific visualization systems tend to have a collection of pre-determined data formats that can efficiently store certain types of classes of data. This has really been a major requirement in the area given the large sizes of the data, and the need to support interactivity.

Take for example the data sets supported by VTK, one of the most widely used and comprehensive visualization toolkits. VTK supports a collection of structured and unstructured data sets. (We are not being comprehensive here, and we refer the extensive documentation of VTK for details.) Structured data sets are n-dimensional matrices, mostly images and volumes. For these data sets the topology is intrinsic, and many basic data operations (e. g., cell location, data look up) are very efficient. Unstructured data sets, where the nodes and cells need to be explicitly defined come in multiple flavors, mostly due to efficiency reasons. A triangulated surface is essentially a version of a 2D surface embedded in 3D space, and it is an example of an unstructured grid, since it requires explicit cell topology information.

Although in principle scientific visualization systems can use existing database technology, the reality is that they implemented all their own (rudimentary) data models, file formats, and metadata management. In the late 1990s, in order to handle ever larger data sets,[1] techniques from the database community started being used in visualization systems. Overviews of those efforts have been given in courses and tutorials by Cox and Ellsworth [8] and Silva et al. [21].

It is interesting to look at the internal processes and data structures used by visualization systems. Scientific visualization systems have traditionally adopted a "data flow" model, where data manipulation operations are represented by linked modules with typed inputs and outputs. These modules define the computations to transform data, and eventually generate pictures. Although conceptually simple, executing such networks gets complicated, in particular when data are large. Optimizations are necessary to avoid wasteful computations under the dataflow model, and caching and parallelism are often used (see, e. g., Bavoil et al [1] and Childs et al [6]). Data sets have temporal components and are so large as to require streaming and out-of-core processing, which further complicate efficient execution of the pipelines. In order to support interactivity, level-of-detail algorithms, where approximate data flows through the module network, need to be supported [22].

While database systems could support many data management tasks for visualization system, such as metadata management, indexing, caching, etc., they have not been widely used. Part of the problem comes from the lack of database expertise by visualization experts. Also, there appears to be a "gap" in functionality needed to efficiently implement visualization algorithms that work on data stored in a database system. For example, support for efficient, vendor-neutral description of multidimensional geometry and related fields needs to be available, as well as for primitives that support level-of-detail (approximate) queries in rendering algorithms.

2.2 InfoVis Systems

Popular infovis toolkits such as Prefuse [12] or the InfoVis Toolkit [9] implement an in-memory database as their main data structure. They transform data tables into visual tables, containing mostly graphical attributes—such as geometric shape, position, color— and render them on the screen.

In InfoVis applications, visual tables are used manage rendering. These in-memory data tables are column-oriented, so adding or removing a column is very cheap and memory access is optimized when few columns are read out of several, which is typical in visualization. The performance requirements are stringent since these tables are used to render the visualization at an interactive rate, typically around 60-100 times a second (60-100 Hz). In current implementations, the data table query is not the bottleneck for rendering large data sets, instead, the bottleneck is the graphics engine. The database constraints are not only in the throughput but also on the latency, because the rendering should be done in sync with the refresh rate of the screen—so the visual table data must be available in main memory all the time.

[1] One article traces the first use of the term "Big Data" to a visualization paper by Cox and Ellsworth.

There are several reasons why the InfoVis community would like to rely on standard database engines instead of the ones they implemented: the services offered by infovis in-memory databases are very rudimentary compared to the standard database services, they need to access standard databases, forcing infovis developers to use two slightly different programming interfaces, which is cumbersome and more complex. However, there are several reasons why standard databases cannot be used currently. We believe the particular services required by the visualization community would benefit many other communities and they may also raise interesting challenges to the database community.

Even if standard database technology would provide optimized library to access databases from regular applications—which they often do—there is currently no way to specify what portion of a queried table should remain in main memory so as to fulfill the real-time constraints. The data types supported by existing databases is not suitable for visualization for two reasons. First, SQL column types need to encode data semantics to allow the selection of good default visual representations. Second, visualization needs to manage geometry in visual tables, and there is currently no support for that. The GIS community has defined some types of standard geometries for their needs, but they have been designed to encode exact 2D topology for mapping management instead of compact geometry for fast rendering. Better support for geometry would be important, both in 2D and 3D, in particular to manage box queries (e. g., what are the objects that are visible in a given box?).

Notification is used extensively in visualization to support linked views and brushing. When data tables are changed, the related visual tables are recomputed, and when the visual tables are changed, the screen is updated. These modifications can come from data updates but are more frequently caused by interactions. For example, clicking on an item on screen will select an object through a selection column in the visual table. This selection will also create a new graphical entity to highlight the selected item, which will be rendered again on screen. The selection can also be managed at the data table level, which is very common when several visual tables are managed for a single data table, i. e., the data table is visualized with multiple visual representations (e. g., showing a spreadsheet view and a scatterplot view). InfoVis databases provide a simple "trigger" mechanism that perform this notification. Implementing a notification mechanism on top of a regular database is complicated and not portable. Client-side notification should be a standard service.

2.3 Visual Analytics

Combining visualization with interactive analysis to explore large data set is the goal of Visual Analytics. The need to support analytical computations creates additional challenges. For example, an infovis system such as Prefuse can load a table from an flat-file format it supports (e. g., a CSV table), pass it to a clustering module to compute groups of items (e. g., using the Weka machine learning toolkit), reduce the table by aggregating the groups and visualize the results (e. g., using a scatterplot visualization). The standard method to implement this pipeline is to copy the content of a Prefuse table into the data model of the clustering program, then to read back in Prefuse the resulting table with one column added, containing a cluster number for each row to be able to project and visualize it. Two data copy operations are thus needed, usually through files.

To address some of these issues, we have designed an abstraction on top of in-memory data tables in a system called Obvious [10]. We have implemented several bindings to check its generality, applicability, and measure its overhead. As we expected, abstracting-out the data model of infovis and data-mining toolkits allows combining very efficiently all these software packages, cutting down the overhead of memory copy whenever possible. The services offered by the Obvious toolkit are a subset of the services we expect from a modern database. Extensive cache management to move data in the memory of the applications that will need it, notification at the client side to manage propagations of value changes, and chains of computations for dynamic data. Propagating value changes from cache to cache is an extremely important operation for VA. In some cases, the visualization and analysis modules can share the same memory (e. g., Java Virtual Machines memory) but, in more realistic settings, the memory cannot be shared because some applications use a specific memory layout that is not abstracted, or because the application needs to run on a different computer (e. g., in the Cloud). In these

cases, the data should move from one memory to the other, using a distribution mechanism. Currently, this communication needs to be managed explicitly by each VA application or, if a distributed database is used, through the database. However, the work consisting in collecting the data changed from one memory to the other, when managed by the VA application, is sub-optimal. Indeed, if the database manager was managing a replicated cache, it would know what portion of the data has been altered and would only move that part. Optimizing that smart distribution is beyond the skills of a VA application programmer. However, from inside the database engine, it looks a lot like database replication and is well understood—though complex to implement correctly.

3 Workflows for Visual Analysis

Visualization systems are just components required for Visual Analytics. Combining computation tools, visualization, and extraction/selection from databases is necessary and can be implemented using ad-hoc tools or more generic *workflow* systems. We report on two of them to discuss the requirements they have on data management services.

3.1 VisTrails

VisTrails (www.vistrails.org) is an open-source provenance management and scientific workflow system that was designed to support the scientific discovery process. VisTrails provides unique support for data analysis and visualization, a comprehensive provenance infrastructure, and a user-centered design. The system combines and substantially extends useful features of visualization and scientific workflow systems. Similar to visualization systems [15, 17], VisTrails makes advanced scientific visualization techniques available to users allowing them to explore and compare different visual representations of their data; and similar to scientific workflow systems, VisTrails enables the composition of workflows that combine specialized libraries, distributed computing infrastructure, and Web services. As a result, users can create complex workflows that encompass important steps of scientific discovery, from data gathering and manipulation, to complex analyses and visualizations, all integrated in one system.

Whereas workflows have been traditionally used to automate repetitive tasks, for applications that are exploratory in nature, such as simulations, data analysis and visualization, very little is repeated—change is the norm. As a user generates and evaluates hypotheses about data under study, a series of different, albeit related, workflows are created as they are adjusted in an iterative process. VisTrails was designed to manage these rapidly-evolving workflows. Another distinguishing feature of VisTrails is a comprehensive provenance infrastructure that maintains detailed history information about the steps followed and data derived in the course of an exploratory task [20]: VisTrails maintains provenance of data products (e. g., visualizations, plots), of the workflows that derive these products and their executions. The system also provides extensive annotation capabilities that allow users to enrich the automatically captured provenance. This information is persistent as XML files or in a relational database. Besides enabling reproducible results, VisTrails leverages provenance information through a series of operations and intuitive user interfaces that aid users to collaboratively analyze data. Notably, the system supports reflective reasoning by storing temporary results, by providing users the ability to reason about these results and to follow chains of reasoning backward and forward [18]. Users can navigate workflow versions in an intuitive way, undo changes but not lose any results, visually compare multiple workflows and show their results side-by-side in a visual spreadsheet, and examine the actions that led to a result [20, 1]. In addition, the system has native support for parameter sweeps, whose results can also be displayed on the spreadsheet [1].

VisTrails includes a number of components that might be best supported directly in a database management system. For instance, VisTrails contains a number of abstractions for files, which are supposed to make it easier to abstract the file type, location (local or Web), and to perform version management. One example is the "persistent file" concept[16], which enables input, intermediate, and output files to be managed through a

version control system (e. g., git). The VisTrails workflow execution engine is user extensible, and it is able to support caching and complex dependencies. In an ideal world, the system would build on state kept by a database. One step towards the use of database technology is a recent addition to the system where it is now possible connect to a provenance-aware relational database [7]. When used correctly, all this functionality nicely complements VisTrails workflow provenance to provide complete, end-to-end provenance support.

3.2 EdiFlow

EdiFlow is an experimental workflow system designed for VA [2] with a focus on reactive computation. A workflow is specified over a database as a set of modules computing the values of output tables from input tables. It is reactive because when a table is modified, the recomputation of the dependent modules is triggered, changing some output tables that can be the input tables of other modules, until all the output tables are recomputed. An important issue for EdiFlow is to avoid useless computations by providing details to modules of what changed in its input tables (the changes to a table are called its delta) so that it can optimize the recomputations. The changes are supposed to arrive by chunk and not one item at a time so the reconciliation are typically managed within transactions. However, since changes can occur at any time, modules can be notified that some input value has changed before their computation is finished. In that case, modules can choose various strategies according to their semantic and implementation. For example, they can stop immediately their computation to restart them from scratch at the next recomputation with the new values. They can also decide to reconcile the new values right away. All these mechanisms are meant to handle database changes asynchronously. These changes can occur for two reasons: either because some data source is changing (e. g., dynamic data is coming in), or because some user has changed values during an interaction with the system. This is the case when EdiFlow is connected to a visualization system and an operation has been performed, such as a user selecting several visualized items, that change the values of a "selection" column in a visual table, that in turn triggers a recomputation of the visualization. While VisTrails manages very effectively the interactive management of scientific workflows, EdiFlow's strength lies in its ability to perform continuous computation of dynamic data.

The heart of EdiFlow relies on two mechanisms that are not standard in databases: notification management (or *client triggers*) and isolation of table values for module computation. The first mechanism is required to keep track of which module should be executed when tables are changed. Each module using an input table that is modified is flagged for execution or recomputation. In addition, for recomputation, each module should be provided the delta, which is maintained by EdiFlow using the trigger mechanism. The second mechanism is required to avoid input tables from a module being run, to be changed spuriously by another module. We have to manage timed values in EdiFlow databases to implement this isolation.

4 Use Cases / Example Apps

We give two very different examples to help understand VA and its data management needs.

4.1 Climate data analysis (UV-CDAT)

Climate scientists have made substantial progress in understanding Earth's climate system, particularly at global and continental scales. Climate research is now focused on understanding climate changes over wider ranges of time and space scales. These efforts are generating ultra-scale data sets at very high spatial resolution. An insightful analysis in climate science depends on using software tools to discover, access, manipulate, and visualize the data sets of interest. These data exploration tasks can be complex and time-consuming, and they frequently involve many resources from both the modeling and observational climate communities.

Because of the complexity of the explorations, the number of tools, and the amount of data involved, database support, including provenance, is critical to support reproducibility and allow scientists to revisit existing com-

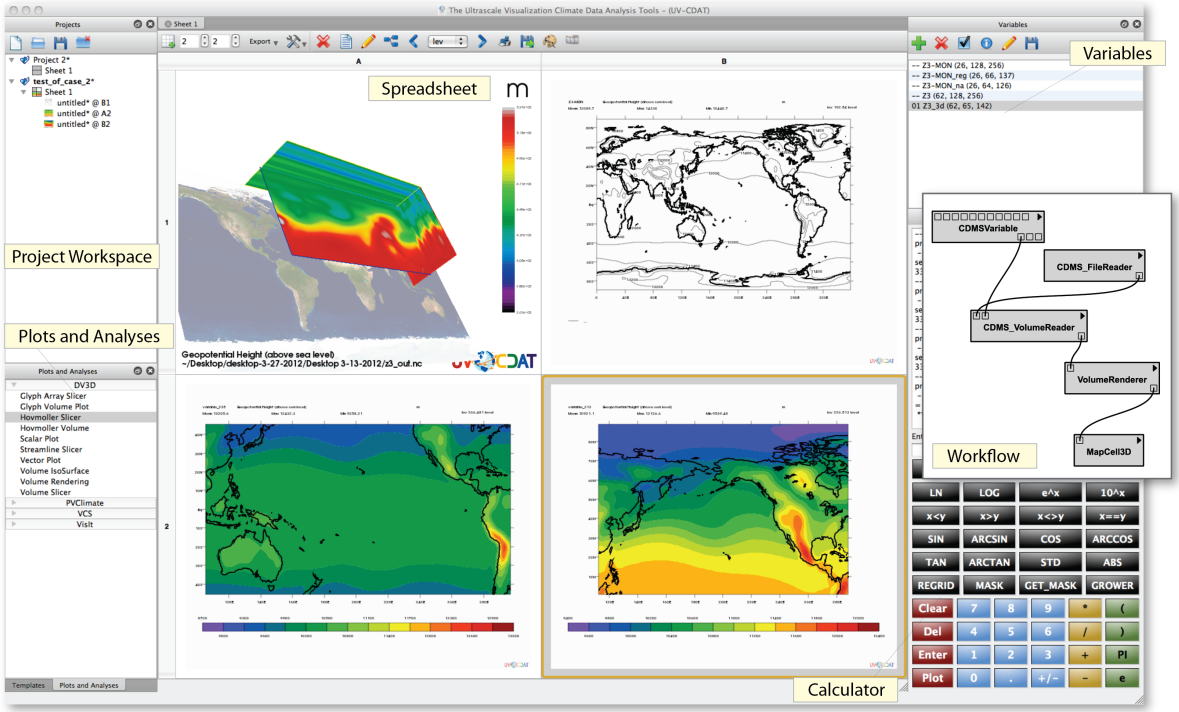


Figure 1: UV-CDAT GUI. Spreadsheet (middle), Project View (top left), Plot View (bottom left), Variable View (top right), and Calculator (bottom right). The system aims to streamline visual data analysis by the use of advanced workflow and provenance support. Image courtesy of Emanuele Santos.

putational pipelines, and more easily share analyses and results. In addition, as the results of this work can impact policy, having provenance available is key to support decision makers. UV-CDAT [19] is a provenance-aware visualization system aimed at supporting large-scale climate data analysis. It uses the VisTrails SDK, which brings data management and provenance capabilities (see Figure 1).

To highlight how UV-CDAT might be used, we use a simplified example from Jerry Porter, a scientist at NASA. The scientist is looking at data from paleoclimate runs on the CCSM3. He wants to determine if the variance of the DJF (December-January-February average) 500 hPa heights (the level of the 500 millibar pressure surface) changes from two different paleoclimate climate simulations. This should give an indication of the changing location of storm track and could be a test of what happens to extratropical storm tracks in a warming earth. He will also need to perform the same analysis for many different periods in the past. The list of steps performed in the analysis are the following: 1) Data discovery: The metadata for the daily model output from the model runs are examined to find the variables. 2) Select a region of interest. For example, the West Coast of the US. 3) Pick a variable and run the variance calculation on the time dimension. 4) Save the data. 5) Plot a 3D Hovmoller diagram (latitude, longitude, time) using DV3D to see the time variation of the geopotential height. 6) Slice the data to examine the region of interest. 7) Plot 2D maps of the subregion, fix the color bar, overlay contours—experiment with other fields to overlay (e. g., moisture flux).

Normally, this process needs to be performed for several models, and the models compared. With existing tools, there is a huge reliance on “files” and naming conventions, and a lot of the meta data is not stored in interchangeable formats. Integration with database, workflow, and provenance systems would streamline the analysis and visualization process and make scientists more efficient.

4.2 Wikipedia Analysis

WikipediaVis [5] is a system to visually present indicators of quality for Wikipedia articles to help assess their trustworthiness. Computing some of these indicators is complex and time consuming. Since the Wikipedia foundation does not maintain computed indices, we decided to compute them on one of our machines and provide a web service to give access to them. We have designed a reactive workflow system to carry the computation task: WikiReactive [4]—the precursor of EdiFlow dedicated to Wikipedia. It computes a cascade of tables related to Wikipedia articles. All these computations are designed to enrich Wikipedia pages with analytical measures to help quickly assess their maturity, known to be correlated to their quality.

At the beginning of the cascade, a program polls Wikipedia to collect and store in our database the list of articles that need processing. Then, the workflow selects one of these articles, fetching its new contents from Wikipedia to compute several measures. For the sake of simplicity, let's assume we only want to compute and maintain, for each character of an article, the user who entered it. This information is very useful because it is the base of the computation of users quality and, accordingly, can be used to show what parts of an article are more trustworthy than others. The first step of our cascade consists in computing the differences (diffs) between the previous and the new version of the article. Other steps involve collecting per user statistics (e. g., number of edits, number of characters added taken from the diffs, etc.), computing the user-per-character table, then computing the real contributors of the article.

This workflow has been designed to gather data and compute measures dynamically and reactively when the pages change. It can be enriched with new modules if other measures become of interest for us, such as natural language analysis or sentiment analysis, thanks to our modular reactive architecture. This process can be seen as a step-wise enrichment of the primary data and is crucial to VA to support interactive search and navigation. For example, the computed diffs allow to quickly find who has inserted a specific word in a Wikipedia article, tremendously shortening the time to find authors of copyright infringements, but also to quickly look at the profile of user editions. This exploration would take hours if it had to be recomputed on the fly.

The data management issues in this application are similar to those faced with EdiFlow: *client triggers* and *isolation*. The main message is that VA applications need reactive workflows to compute important statistics, derive values, and analytical information from dynamic data sources.

5 Summary

Visualization has mimicked/reinterpreted database technologies for various reasons. We need to reconcile our software infrastructures for data management with standard databases but this requires some improvements and new capabilities.

Supporting geometric data used by visualization: visualization manages 2D and 3D data, VA needs to move this data across multiple execution environments. Standardizing geometry storage and supporting queries would address a major reluctance of visualization researchers towards using standard databases. GIS extensions already specify some geometry but not with the same semantics. For 2D visualization, the XML/SVG standard would fit the needs and is already well specified. For 3D, more work is needed to define vendor-neutral structures. In the latter case, a major issue would be storage efficiency since 3D geometry, as managed by current visualization applications, can be large.

Supporting geometry at multiple scales is also essential. Just like Google Earth specifies maps at different scales, visualization also needs to retrieve geometry at multiple scales. Geometric queries include intersection, and inclusion. Indexing these objects is a well-known issue in computational geometry and techniques from GIS databases can be reused.

Supporting extensive caching is mandatory to avoid duplicating data structures and maintaining the required computation and rendering speed for visualization structures. Interactive visualization needs high speed and low latency with real-time guarantees. We share the vision of Boncz et al. [3] of the database as a shared memory

with smart caching to use the existing hardware optimally. This vision differs from the classical database with explicit load and store operations. If in-memory caching is supported, *fast distribution of cached data at transaction time* should also be supported to propagate changes across applications. This would allow computation modules to exchange computed results quickly without having to implement their own communication mechanism when the database already knows how to distribute data. With extensive caching, *smart disconnection and reconnection management* should be supported to allow applications to run while connected but also in isolation.

For VA, *long transaction models* should be supported. More work is needed to define a minimum set of models but the current model where a transaction either succeeds or fails is too simple. Failed transactions should be resumable after some reconciliation of data, and computations should be notified of concurrent updates to possibly address conflicts earlier than at the end of long transactions.

Version control should be provided at the database level for provenance management and concurrent computations in workflows. Both Oracle and IBM have recently introduced temporal capabilities that keep track of the complete history of tables, providing support for some of our needs.

Asynchronous operations are required to analyze large data sets while providing continuous feedback to users. This may require trading query quality for speed as described in [14]. Additionally, during data exploration, *most of the queries submitted are extensions or restrictions of recent queries*, offering potential speedups for fast approximate queries. *Mechanisms to explicitly stop, refine, extend, resume queries* would allow more user control over the system's behavior and faster initial results.

Approximate/Partial aggregate queries would be extremely valuable for VA, as described in [11]. The proposed system continuously returns partial values of average queries while the query is being run on the whole database, allowing users to quickly see the results applied to a growing sample of data until it completes. Coupled with the previous mechanism for controlling the query, this mechanism would offer interactive continuous feedback, which is essential for VA.

While database vendors already support some of the requirements we discussed, other requirements need further research. We believe that this research will enrich both the Visual Analytics and the database fields.

We note that the increasing importance of scientific data sets has caught the attention of the database community, as evidenced by the work of Howe and Maier [13]. Recently, we are starting to see more work at the interface of these areas, and we expect that in the future, database and visualization systems will be able to be much more tightly coupled.

Acknowledgments: Silva's work has been funded by the U.S. Department of Energy (DOE) Office of Biological and Environmental Research (BER), the National Science Foundation, and ExxonMobil.

References

[1] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. VisTrails: Enabling interactive, multiple-view visualizations. In *IEEE Visualization 2005*, pages 135–142, 2005.

[2] V. Benzaken, J.-D. Fekete, P.-L. Hémery, W. Khemiri, and I. Manolescu. EdiFlow: data-intensive interactive workflows for visual analytics. In *ICDE 2011*, pages 780–791, 2011.

[3] P. A. Boncz, M. L. Kersten, and S. Manegold. Breaking the memory wall in monetdb. *Commun. ACM*, 51(12):77–85, Dec. 2008.

[4] N. Boukhelifa, F. Chevalier, and J.-D. Fekete. Real-time Aggregation of Wikipedia Data for Visual Analytics. In *VAST 2012*, pages 147–154, 2010.

[5] F. Chevalier, S. Huot, and J.-D. Fekete. WikipediaViz: Conveying Article Quality for Casual Wikipedia Readers. In *PacificVis 2010*, pages 215–222, 2010.

[6] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max. A contract-based system for large data visualization. In *IEEE Visualization 2005*, pages 190–198, 2005.

[7] F. Chirigati and J. Freire. Towards integrating workflow and database provenance: A practical approach. In *International Provenance and Annotation Workshop (IPAW)*. Springer Verlag, 2012.

[8] M. Cox and D. Ellsworth. Managing Big Data for Scientific Visualization. In *ACM SIGGRAPH '97 Course #4, Exploring Gigabyte Datasets in Real-Time: Algorithms, Data Management, and Time-Critical Design*, 1997.

[9] J.-D. Fekete. The InfoVis Toolkit. In *InfoVis 04*, pages 167–174, 2004.

[10] J.-D. Fekete, P.-L. Hémery, T. Baudel, and J. Wood. Obvious: A meta-toolkit to encapsulate information visualization toolkits — one toolkit to bind them all. In *VAST 2011*, pages 89–98, 2011.

[11] D. Fisher, I. Popov, S. Drucker, and M. Schraefel. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *CHI '12*, pages 1673–1682, 2012.

[12] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a Toolkit for Interactive Information Visualization. In *CHI '05*, pages 421–430, 2005.

[13] B. Howe and D. Maier. Algebraic manipulation of scientific datasets. In *VLDB '04*, pages 924–935. VLDB Endowment, 2004.

[14] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou. The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB*, 4(12):1474–1477, 2011.

[15] Kitware. Paraview. `http://www.paraview.org`.

[16] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva. Bridging workflow and data provenance using strong links. In *SSDBM*, pages 397–415, 2010.

[17] Lawrence Livermore National Laboratory. VisIt: Visualize It in Parallel Visualization Application. `https://wci.llnl.gov/codes/visit` [29 March 2008].

[18] D. A. Norman. *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Addison Wesley, 1994.

[19] E. Santos, D. Koop, T. Maxwell, C. Doutriaux, T. Ellqvist, G. Potter, J. Freire, D. Williams, and C. Silva. Designing a provenance-based climate data analysis application. In *International Provenance and Annotation Workshop (IPAW)*. Springer Verlag, 2012.

[20] C. Silva, J. Freire, and S. P. Callahan. Provenance for visualizations: Reproducibility and beyond. *IEEE Computing in Science & Engineering*, 2007.

[21] C. Silva, Y. jen Chiang, W. Corrła, J. El-sana, and P. Lindstrom. Out-of-core algorithms for scientific visualization and computer graphics. Technical report, LLNL UCRL-JC-150434-REV-1, 2003.

[22] H. Vo, J. Comba, B. Geveci, and C. Silva. Streaming-enabled parallel data flow framework in the visualization toolkit. *Computing in Science Engineering*, 13(5):72 –83, sept.-oct. 2011.

Usability, Databases, and HCI

Fei Li and H. V. Jagadish
Univ. of Michigan

Abstract

As usability is becoming recognized as a crucial feature for databases, there is growing reason for us to pay attention to the principles of Human Computer Interaction (HCI). This article explores the interaction of HCI with databases. We can learn from specific HCI areas, such as information visualization, from general HCI principles, such as direct manipulation, and from HCI methodology, such as running good user studies. However, there is much required for database usability that goes beyond HCI.

1 Usability in Databases

Computer professionals have long recognized the importance of organizing data. The core principles of database technology were established early in the history of modern computing. Today, databases are widely used in a broad range of organizations, and are the basis for a vibrant commercial sector.

In spite of these successes, there has been a longstanding question asked by many about why so much data is not in databases. In comparison, consider compilers, another computing technology whose core principles were developed early. There do remain today a few specialized situations in which people choose to write assembler directly and forgo the benefits of compiling a high level language, but these situations are rare. Why do so many choose so frequently to give up the many wonderful benefits provided by databases?

Over the years, attempts have been made to address this question, most notably through object oriented databases. However, the question has not been a central concern of the database field: when there is so much growth and such great need, it is hard to devote attention to the audiences not being served well.

The web has changed the situation dramatically. If you ask a non-technical person today, they will think of the web as the greatest (distributed) information store and of a search engine (like Google) as the way to (organize and) access this information. Structured databases come only much lower down in perceived importance. Furthermore, the web has democratized information, through disintermediation. When travel agents were the only users of a reservation system, they could be trained to learn magic letter codes and unintuitive access paths. But today, most of us make our own travel arrangements, without a human travel agent as an intermediary. The general user population is not willing to learn magic incantations and gets frustrated with unintuitive access paths. This democratization of database access has led to a greatly increased need for usability.

Not surprisingly, usability issues have gained importance in the database community in recent years. Since usability is a central objective of the human computer interaction (HCI) community, it is natural to ask what we can learn from them. This paper examines some HCI accomplishments that relate to databases, and argues that there still remains much that we need to do beyond that.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Data Management, HCI and User Studies

Computers are notoriously logical: they do precisely what they are instructed to do – no more and no less. Since this is not the way that humans interact with one another, and since what computers do can be quite complex, it is not surprising that many humans find it challenging to interact with computers. HCI developed as a field to address this challenge. The focus in HCI has been on how a human can interact with a computing system to get a task done. Since most tasks that a computer can help with are computational in nature, the traditional focus has been on the computation rather than the data. Only recently has there been enough, and complex enough, digital data that information search has itself become an important computational task. In consequence, database systems have become a more important application domain for HCI work.

When evaluating human interaction with computers, the humans in the loop are central – it is usually not possible to evaluate an HCI system without humans! Therefore, user studies, of many different types, have been used widely by HCI researchers [15]. In contrast, most database systems work is concerned with functionality and performance, which can usually be measured directly from the system, without including the user. As a result, user studies have been rare in database systems work. However, this is beginning to change – there is an increasing number of papers published in database venues that now include a user study. As we in the database field start running more user studies, there is much we can learn from others who have run such studies before, and HCI experts in particular. In other words, in addition to the specific research topics that we will discuss below, there is methodological expertise in the HCI community that is likely to be of value to many in the database community. So it worth our effort to learn from them.

The most common type of user study one sees in a database usability paper has the users performing some task using two (or more) alternative systems, say one (or more) baseline and the new system [50, 37, 36]. For each, the study typically measures time to task completion, and may possibly also measure quality of task performance (such as correctness of response). Care has to be taken to balance the two groups, and this includes considerations that computer systems do not often have, such as learning effects and fatigue effects. Furthermore, user studies are considerably more expensive to run than computer system performance studies, so what is evaluated has to be planned with care. In HCI, and in other fields that routinely run user studies, such as many social sciences, there is considerable attention paid to exactly what the users are told, since user knowledge and expectations can bias observed results [15]. In database user studies, such niceties are not often considered. Similarly, in HCI, one may perform an observational study – just recording where the user spent time and where they were confused. There may also be carefully designed (subjective) questionnaires. While some database user studies do look at such points, there is scope to do so in a much more sophisticated manner.

A new methodological wrinkle is due to remotely run user studies on sites such as Amazon's Mechanical Turk [2]. Some "turkers" may be trying to maximize income as a function of time spent, and so may not put in effort to answer questions carefully – such"cheating" is much less likely when the facilitator is physically present as in a traditional study. Many a researcher has been surprised by nonsense study results due to lack of subject effort. For example, a "turker" may choose the first option in each multiple-choice question, just to move on as fast as possible, possibly without even reading the question. A study designer, in response, may build in a small minimum delay before the next question will load, forcing the "turker" to slow down. The "turker" in turn may counter this by opening multiple windows and performing multiple tasks in parallel, still paying little attention to each, but satisfying the minimum time requirement. The design of studies in such an environment is a small sub-field in itself [25], not really core to traditional HCI, but closely related. It is also believed [7] that for many "turkers," even though the intention is not to cheat overtly, if the instructions are not clear or the question asked is confusing, a common path taken is to guess (or choose randomly) and move on, rather than to put in the effort to clarify. Therefore, the usability of the study itself becomes an important requirement for its success.

Of course, user evaluation is only one piece of most HCI work. The bulk of the contribution from HCI is in designing new tools and techniques that improve usability.

3 Understanding and Interpreting Data

Databases contain structured information, and the results produced can be voluminous. It has long been recognized that humans can benefit from better data presentation, particularly when the data to present are voluminous. Edward Tufte's book [47] is a classic. But this work, important though it is, hardly represents the beginning: rather information visualization has been studied for as long as there has been information, and certainly well before the age of computers. The *diagrammatic* map of the London underground was devised by Harry Beck in 1931 [3]. By not remaining faithful to the actual geography, a much more "readable" map of the subway system was developed than had been there before. Such diagrammatic maps are now typical in most subway systems worldwide. Even older than this is Charles Minard's 1869 chart showing the number of men in Napoleon's 1812 march to Russia [1]. This chart has been widely acclaimed for how well it represents multiple variables, including the size of army, the passage of time, and the geography, all in a single compelling graphic.

3.1 Result Presentation

The typical goal for information visualization is "giving to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space". As researchers have thought about how to present large query result sets, they have developed many innovative ways to look at information.

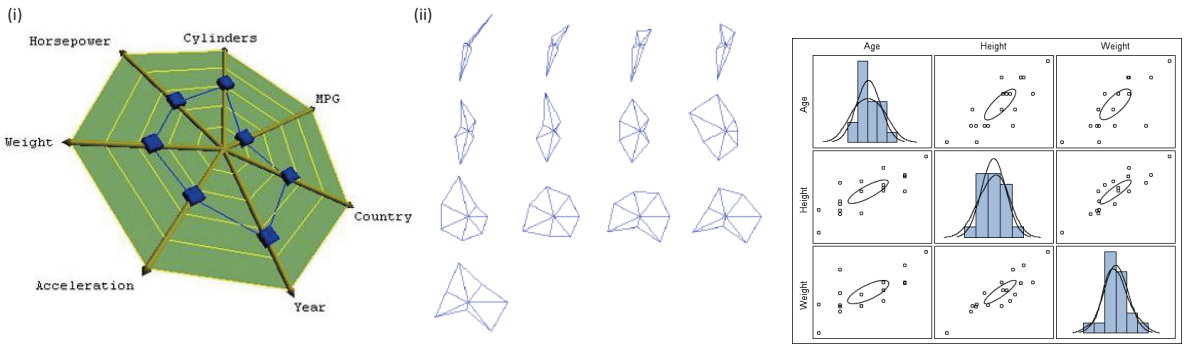
For example, in Star plot [11], each (multi-attribute) data item is shown as a star and all the attributes are represented as equal angular axes radiating from the center of a circle, with an outer line connecting the data value points on each axis. As a result, Star plot is able to clearly present thousands of data items with tens of attributes to the users (Figure 1(a)). Another technique, Scatterplot matrices [24], is designed to show the relationship between different attributes. It enumerates all pairs of attributes, takes each pair of attributes as a projection, maps each projection in one 2D scatterplot, and finally organizes these scatterplots in a scatterplots matrix. By doing this, even if there are many data items in the database, users can clearly observe the correlation between each pair of attributes (Figure 1(b)).

Instead of just listing all data item on screen, some techniques convey the relationship between data items to the users. An example is Narcissus [22], in which the system takes each web page as a node and each hyperlink as an edge, and shows the graph structure of the web (Figure 1(c)).

When displaying these hierarchical (or graph) data, lower levels can quickly get crowded since the number of nodes per level can grow exponentially in a tree (or a graph). To deal with this crowding problem, hyperbolic tree [31] employs hyperbolic space, which intrinsically has "more room" than Euclidean space and clearly visualizes large, wide hierarchical data (Figure 1(d)). Similarly, treemap [42] hierarchically partitions the screen into nested regions and potentially increase the space for showing hierarchical data. Moreover, treemap can also give a good overview of large datasets with multiple ordinal attributes by using a region size to denote an attribute value (Figure 1(e)). Another way to show large graph data is to use interactive distortion techniques, to emphasize part of the data or just to make the space larger. "Overview first, zoom and filter, then details on demand [43]" are the typical processes. For instance, Fisheye Views [18] dynamically show areas of interest larger and with more detail (Figure 1(f)).

Even where the information being searched is not particularly structured, there may be value in presenting the results in a structured way. Search engines group together results from similar sites. Research papers have considered clustering search results based on topic [38]. The crudest examples are for disambiguating homonyms, such as jaguar the animal, the car make, and the football team. But even when there is no such gross disambiguation involved, say given a search for the company "Samsung", we may find it useful to cluster together pages about Samsung products and separately cluster together pages about the Samsung company financials and news mentions.

One way to impose visual structure that has become very popular of late is the *tag cloud* (also sometimes called a *word cloud*). The idea is to choose the most commonly occurring terms (words/tags/...) in a corpus

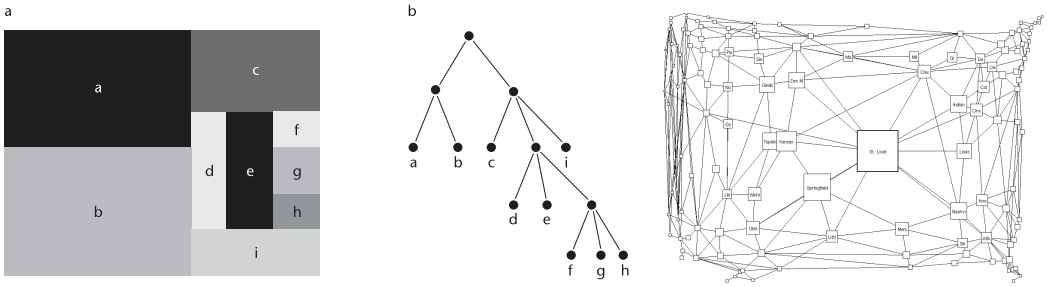


(a) Star plot: (L) The left figure in a is one star glyph, representing seven attributes of a car in which the length in each angular axe shows the value in the corresponding attribute. (R) The right figure shows a group of star glyphs, which take a small space but can potentially show large numbers of star glyphs [12].

(b) Scatterplot matrices: This scatterplot matrix for 3-dimensional data show the relationships between each pair from attribute set {age, weight, and height}.

(c) Narcissus: Representation of complex web structure of a collection of web pages in Narcissus, in which each node is a webpage, each link a reference relationship. The large nodes are indexes into the pages, and the ball-like structure represents the set of cross referenced pages [22].

(d) Hyperbolic tree: This Hyperbolic tree locates its root in the center of the space and all its offsprings in the hyperbolic plane hierarchically, and clearly shows the layout of a tree with 1004 nodes [31].



(e) Treemap: The TreeMap in the left show the hierarchical structure of the tree structured data in the right [42].

(f) Fisheye Views: Fish-eye view of links between major American cities with a focus on St. Louis [39].

Figure 1: Examples in Information Visualization Techniques.

and to show these in a manner that informs the user. Typically, the font size is used to convey frequency of occurrence. The boldness and color of font, as well as placement of words in the cloud, can be used to represent information about the unstructured corpus in a succinct way.

See [10, 41] for a good collection of research works on information visualization. The ManyEyes system [49, 48] has implemented many of the better known information visualization methods.

3.2 Visual Analytics

Crudely speaking, database applications are divided into two types: the first is operational or transactional, and the second is warehousing. Typically, the former do not produce large or complex results. In consequence, much of the focus of information visualization has been on the latter. But the typical consumer of information from a data warehouse is a decision-maker, who is using the displayed warehouse information to better understand and analyze it in support of the decision to be made. It is also well-known that a typical decision-maker does not issue a few queries in one shot, get results and base all decisions on the results of these queries. Rather, there is a sequence of queries, examining data that is surprising from multiple angles, drilling down into outlier aggregates, running what-if analyses, and so on. In short, we have a user task accomplished through a sequence of queries, each of which produces a query result that can be visualized using suitable techniques such as those mentioned in the preceding sub-section. This bigger picture then begs the question why the user has to look at well-presented information and then develop a query in a separate window/interface. Couldn't we short-circuit this, have the user directly interact with the visualized information, and thereby analyze data from the warehouse more effectively? The HCI principle of direct manipulation certainly suggests this

The epiphany above gave rise to the field of *visual analytics* [45], also sometimes referred to as *visual data mining*. This is also central to the commercially sold Tableau system, based on VizQL [21]. But let's step back for a moment from the picture painted in the preceding paragraph, and think about what the user is really trying to do. The user wishes to find interesting patterns or items worthy of attention, and is doing so through a process that involves a repeated cycle of visualization and interactive commands. In other words, the human and the computer system are symbiotically solving a problem: the human is looking for patterns in the visually presented data and also thinking about what presentations may be most informative, while the system is crunching through large quantities of data to develop informative visual presentations. Developing such a system requires the developer to think about how to make the human most effective at finding patterns of interest. In contrast, the traditional data mining problem is to develop a system that can itself find patterns of interest. Where the pattern of interest is easy to define precisely, there is no question that the system can do very well. But where, as is often the case, there could be many patterns of interest or patterns are hard to define, then the flexibility of the human brain is a tremendous advantage, and one should expect visual analytics to be far more powerful than traditional computer-driven analytics.

4 Query Specification

The preceding discussion primarily dealt with the presentation of information from a database, and indeed this is where the bulk of the work is, in dealing with information sources. However, before we get to look at the answer, we have to pose the question and get the expected query results. In a traditional database system, this is not easy. Standard query languages, like SQL or XQuery, while expressive and powerful, require the users to learn both the database schema and how to compose the exact query statements. Since this learning curve is often too steep for naive users, usable query mechanisms are in dire need. Ideally, a perfect usable query system should be like a private intermediary agent who is a database expert and works tirelessly for the end-users. The end-user thinks of what she wants to query and expresses this thought in her own idiosyncratic way to the intermediary agent. Because of the knowledge gap, the intermediary agent needs to figure out what the user exactly wants to query.

In the old days, this intermediary agent may have been a staffer in a computer support group. Today, the delay of going through such a staffer is usually unacceptable, even if the cost were acceptable. Therefore, our desire is to build a computer system that can perform this role of intermediary agent. In this section, for each query mechanism, we first describe some state-of-the-art works in the intermediary agent perspective. Then we discuss the gap between these state-of-the-art works and the ideal usable system in the same query mechanism, and finally try to figure out possible ways to improve these existing works.

4.1 Visual Interfaces

Forms are the traditional way for naive users to query a database. The designer of the interface carefully designs one or more forms by making an educated guess of which data the users would be interested in and how these desires may be mapped to query statements in the database. Then she gives these forms to the intermediary agent, who converts the form into a DBMS query. Forms work very well when the query logic of the end-users can be expressed by these forms. But, what if the end-user wants a query that cannot be expressed by the existing forms? Since the designer cannot design all forms for all the end-users, she could give some form generation rules to the intermediary agent [27, 28], letting the intermediary agent generate more forms and hoping some of these forms can express the expected query logic. The intermediary agent may even be able to take keywords as a preliminary input, gives the forms related to these keywords to the end-users [13], and hopes one of these forms can exactly express the expected query logic. The ideal form-based agent should always be able to provide one form which can express the expected query logic of the end-users. That is, the user tells the agent what information she needs in her own way and the agent generates the form on the fly. Getting closer to this ideal remains a current topic of research.

There are also many endeavors in non-form visual query mechanisms that enable end-users to compose natural, generic query logics. These works began in 1975 with QBE [51], which allows users to query a database by creating example tables in the query interface. The agent then translates them into standard query language, and finally executes it on the database. After QBE, many other visual query languages have been proposed in academia, like QBT [40], XQBE [9], MIX [34], Xing [16], Kaleidoquery [35]. In industry also, visual query specification (and visual database design) are commonly supported. For example, Microsoft Access supports QBE-based query over relational databases and IBM uses Visual XQuery Builder to support visual queries over XML databases. While expressive, these methods require the users to specify how these data are structured and where to look for the information they desire, which makes them unsuitable for totally naive users. An ideal agent should allow the user to input the examples in the user's paradigm, not in the system's paradigm, then analyze what the users want to query by these examples (with some feedback for confirmation if needed), and finally translate these examples into standard query statements.

4.2 Text Interfaces

Natural language interfaces for databases (NLIDBs) have a long history [6]. Most NLIDB systems, like Lunar [17], parse the query into a parse tree and then map the parse tree into a database query expression. Also, some interactive NLIDB query systems [29, 46] are proposed to guide the query formulation interactively and adaptively. However, most of these early works suffer from both domain dependency and database dependency, which rely heavily on domain knowledge and manually created adaptation rules for each database.

Today, the most widely used query mechanism for naive users is the keyword-based query interface. Without the requirements of any schema information or any programming skills, the agent only asks the users for some keywords and then returns the data that are relevant to these keywords. In general, the relevant data is a group of closely inter-connected tuples, according to foreign-primary key relationship (in relational databases [8, 5, 23]) or parent-children relationship (in XML databases [14, 20]), that contain all the keywords.

However, many user needs cannot be satisfied with just any relevant data provided by the agent. . Unfortunately, a bag of keywords often does not capture enough information about the user's information need, and the system's guess at completing this need is either wrong, or too non-specific [26].

To make the search more expressive and effective, some recent works try to figure out the semantic information in the keywords (e.g. SQAK [44]) or enable the users to specify some semantic information in addition to keywords (e.g., Schema-Free XQuery [33], natural language [32], query based on schema summary [50]). SQAK [44] extracts the aggregation information from flat structured keywords. But this work falls far short of removing all the ambiguity in keywords. Schema-free XQuery [33] integrates keyword search functionality into XQuery as built-in functions and enables users to query XML documents based on partial knowledge they may have about the database schema. Furthermore, NaLIX [32] provides a generic natural language query interface, instead of keywords, to XML database. Since natural language itself can convey semantic information, it reduces the uncertainty resulting from the intrinsic ambiguity of flat keywords. Another work, query schema summary [50], generates schema summary and supports queries on this simple schema summary rather than on the original complex schema.

5 Conclusion

As the use of databases is "democratized," in that end-users interact with them directly rather than being mediated by MIS staff, the importance of usability in databases is growing. There is much that we as a community could do in this regard, and much that we can learn from HCI. However, HCI alone is not going to solve all our usability problems: there is now a bonafide sub-field of database usability, which squarely deals with databases while drawing inspiration from work in HCI. This article provides an overview of the issues.

Since this article is written for a database audience, the bulk of it is devoted to the impact of HCI on databases. This does not imply that there isn't impact in the other direction. It is now possible to record every action performed by the user, and the time taken to perform these actions. With a little bit of instrumentation, one can go even further, to track eyeball movements, for example. This gives rise to a significant size database from a single user session. Analyzing patterns in this database has already been demonstrated to be of value in developing better interfaces in a range of applications from web search engines [19] to video games [30]. In short, there is much that database technology can contribute to HCI, just as in the reverse direction.

6 Acknowledgment

This work was supported in part by NSF under grants IIS 0741620 and IIS 1017296.

References

[1] All graphic works of charles joseph minard: http://www.datavis.ca/gallery/minbib.php.

[2] Amazon mechanical turk: http://aws.amazon.com/mturk/.

[3] London subway: http://diagrams.org/fig-pages/f00022.html.

[4] Visualizing higher dimensional data: http://www.mathworks.com/products/demos/statistics/mvplotdemo.html.

[5] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16, 2002.

[6] L. Androutsopoulos. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, 1:29–81, 1995.

[7] M. Bernstein. Personal communication, 2012.

[8] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE*, pages 431–440, 2002.

[9] D. Braga, A. Campi, and S. Ceri. Xqbe (xquery by example): A visual interface to the standard xml query language. *ACM Trans. Database Syst.*, 30(2):398–443, June 2005.

[10] S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[11] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. Graphical Methods for Data Analysis. *The Wadsworth Statistics/Probability Series. Boston, MA: Duxury*, 1983.

[12] W.-Y. Chan. A survey on multivariate data visualization. *Technical Report*, 2006, HKUST.

[13] E. Chu, A. Baid, X. Chai, A. Doan, and J. Naughton. Combining keyword search and forms for ad hoc querying of databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 349–360, New York, NY, USA, 2009. ACM.

[14] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. Xsearch: A semantic search engine for xml. In *VLDB*, pages 45–56, 2003.

[15] J. F. Dumas and J. C. Redish. *A Practical Guide to Usability Testing*. Greenwood Publishing Group Inc., Westport, CT, USA, 1993.

[16] M. Erwig. A visual language for xml. In *Proceedings of the 2000 IEEE International Symposium on Visual Languages (VL'00)*, VL '00, pages 47–, Washington, DC, USA, 2000. IEEE Computer Society.

[17] W. W. et al. *The Lunar Sciences Natural Language Information System: Final Report.* Bolt Beranek and Newman Inc., Cambridge, MA, 1972.

[18] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '86, pages 16–23, New York, NY, USA, 1986. ACM.

[19] Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 417–420, New York, NY, USA, 2007. ACM.

[20] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *SIGMOD Conference*, pages 16–27, 2003.

[21] P. Hanrahan. Vizql: a language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 721–721, New York, NY, USA, 2006. ACM.

[22] R. J. Hendley, N. S. Drew, A. M. Wood, and R. Beale. Case study: Narcissus: visualising information. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, INFOVIS '95, pages 90–, Washington, DC, USA, 1995. IEEE Computer Society.

[23] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, pages 670–681, 2002.

[24] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[25] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS*, 17(2):16–21, Dec. 2010.

[26] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 13–24, New York, NY, USA, 2007. ACM.

[27] M. Jayapandian and H. V. Jagadish. Automating the design and construction of query forms. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 125–, Washington, DC, USA, 2006. IEEE Computer Society.

[28] M. Jayapandian and H. V. Jagadish. Automated creation of a forms-based database query interface. *Proc. VLDB Endow.*, 1(1):695–709, Aug. 2008.

[29] E. Kapetanios and P. Groenewoud. Query construction through meaningful suggestions of terms. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems*, FQAS '02, pages 226–239, London, UK, UK, 2002. Springer-Verlag.

[30] J. H. Kim, D. V. Gunn, E. Schuh, B. Phillips, R. J. Pagulayan, and D. Wixon. Tracking real-time user experience (true): a comprehensive instrumentation solution for complex systems. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 443–452, New York, NY, USA, 2008. ACM.

[31] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In I. R. Katz, R. L. Mack, L. Marks, M. B. Rosson, and J. Nielsen, editors, *CHI*, pages 401–408. ACM/Addison-Wesley, 1995.

[32] Y. Li, H. Yang, and H. V. Jagadish. Nalix: an interactive natural language interface for querying xml. In *SIGMOD Conference*, pages 900–902, 2005.

[33] Y. Li, C. Yu, and H. V. Jagadish. Schema-free xquery. In *VLDB*, pages 72–83, 2004.

[34] P. Mukhopadhyay and Y. Papakonstantinou. Mixing querying and navigation in mix. In R. Agrawal and K. R. Dittrich, editors, *ICDE*, pages 245–254. IEEE Computer Society, 2002.

[35] N. Murray, N. Paton, and C. Goble. Kaleidoquery: a visual query language for object databases. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '98, pages 247–257, New York, NY, USA, 1998. ACM.

[36] L. Qian, M. J. Cafarella, and H. V. Jagadish. Sample-driven schema mapping. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 73–84, New York, NY, USA, 2012. ACM.

[37] L. Qian, K. LeFevre, and H. V. Jagadish. Crius: user-friendly database design. *Proc. VLDB Endow.*, 4(2):81–92, Nov. 2010.

[38] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina. Clustering the tagged web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 54–63, New York, NY, USA, 2009. ACM.

[39] M. Sarkar and M. H. Brown. Graphical fisheye views. *Commun. ACM*, 37(12):73–83, Dec. 1994.

[40] A. Sengupta and A. Dillon. Query by templates: a generalized approach for visual query formulation for text dominated databases. In *Proceedings of the IEEE international forum on Research and technology advances in digital libraries*, IEEE ADL '97, pages 36–47, Washington, DC, USA, 1997. IEEE Computer Society.

[41] B. Shneiderman and B. B. Bederson. *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[42] B. Shneiderman and C. Plaisant. *Treemaps for space-constrained visualization of hierarchies*. http://www.cs.umd.edu/hcil/treemap-history/index.shtml, 1998-2009.

[43] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.

[44] S. Tata and G. M. Lohman. Sqak: doing more with keywords. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 889–902, New York, NY, USA, 2008. ACM.

[45] J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Comput. Graph. Appl.*, 26(1):10–13, Jan. 2006.

[46] A. Trigoni. Interactive query formulation in semistructured databases. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems*, FQAS '02, pages 356–369, London, UK, UK, 2002. Springer-Verlag.

[47] E. R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, USA, 1986.

[48] F. B. Viégas and M. Wattenberg. Shakespeare, god, and lonely hearts: transforming data access with many eyes. In *JCDL*, pages 145–146, 2008.

[49] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. M. McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1121–1128, 2007.

[50] C. Yu and H. V. Jagadish. Querying complex structured databases. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1010–1021. VLDB Endowment, 2007.

[51] M. M. Zloof. Query-by-example: the invocation and definition of tables and forms. In *Proceedings of the 1st International Conference on Very Large Data Bases*, VLDB '75, pages 1–24, New York, NY, USA, 1975. ACM.

Big Data Methods for Computational Linguistics

Gerhard Weikum, Johannes Hoffart, Ndapandula Nakashole,
Marc Spaniol, Fabian Suchanek, Mohamed Amir Yosef

Max Planck Institute for Informatics
Saarbruecken, Germany
E-mail: weikum@mpi-inf.mpg.de

Abstract

Many tasks in computational linguistics traditionally rely on hand-crafted or curated resources like thesauri or word-sense-annotated corpora. The availability of big data, from the Web and other sources, has changed this situation. Harnessing these assets requires scalable methods for data and text analytics. This paper gives an overview on our recent work that utilizes big data methods for enhancing semantics-centric tasks dealing with natural language texts. We demonstrate a virtuous cycle in harvesting knowledge from large data and text collections and leveraging this knowledge in order to improve the annotation and interpretation of language in Web pages and social media. Specifically, we show how to build large dictionaries of names and paraphrases for entities and relations, and how these help to disambiguate entity mentions in texts.

1 Introduction

Methods for data analytics are relevant for all kinds of information, including text. Although we live in the era of Big Data, Linked Data, Data-Driven Science, and the Data Deluge, for humans the most informative contents on the Internet still is in *natural-language text*: books, scholarly publications, news, social media, online communities, etc. Making sense of natural language falls into the field of computation linguistics (CL). Semantics-focused tasks in CL include question answering (e.g., of the kind addressed by IBM Watson), word sense disambiguation (i.e., mapping ambiguous words such as "plant" or "star" to their meanings) co-reference resolution (e.g., mapping pronouns to a preceding noun phrase), semantic role labeling, paraphrasing and textual entailment, automatic summarization, and more [2, 21, 28, 36, 39, 40].

Data scarceness: The models and methods that computational linguistics has developed for these tasks over several decades benefit from data collections and text corpora, but usually require human-quality markup and ground-truth annotations, compiled and curated by experts (incl. quality measures such as inter-annotator agreement). CL refers to these assets as resources, as opposed to the raw sources. As human effort is the bottleneck, the available resources tend to be fairly small. A corpus with thousands of short documents (e.g., news articles), ten thousands of sentences, and millions of words is considered large. For tasks that need only unlabeled data, such as statistical machine translation, Web corpora can be harnessed and have indeed strongly influenced the state of the art. However, the semantic tasks mentioned above rely on deeper resources with

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

fine-grained labeling. Note that crowdsourcing is not a viable alternative for expert-quality annotations. Entity recognition and disambiguation, for instance, often requires thoughtfulness that goes beyond what the typical mturk worker delivers. For example, the sentence "Berlin talks to Paris about the Euro problem with Greece and Spain" should yield four entities: German government (not the city of Berlin, a case of metonymy), French government, financial crisis in Greece, financial crisis in Spain.

Time for big data: This situation has changed in the last few years. Starting in 2006, projects like DBpedia [3], freebase.com, WikiTaxonomy [42], and YAGO [47] have constructed huge knowledge bases (KBs) of entities (people, places, etc.), their semantic classes (e.g., musicians, waterfalls, love songs, etc.), and relationships between entities (e.g., worksFor, wonPrize, marriedTo, diedIn). To this end, YAGO tapped into knowledge-sharing communities like Wikipedia, and integrated the data derived there with existing CL resources like the WordNet thesaurus [12] as a semantic backbone. The resulting KBs are big-data assets that combine the rigor and quality of traditional resources with the wealth and scale of automatically harvested Web sources. This trend is still ongoing: KBs keep growing, specialized KBs are created, KBs accelerate the CL-style annotation of large corpora, and many of these resources are semantically interconnected at the entity level in the Web of Linked Data [18]. The following two examples illustrate how such big data assets can contribute to advanced tasks in computational linguistics.

Example 1: Consider a question answering scenario where a user asks: "Which composer from the eternal city wrote the score for the Ecstasy scene?", perhaps with the additional cue "..., which was later covered by a classical string player and a metal band". There is plenty of data available to retrieve the answer (in imdb.com, freely available KBs, or Web pages), but the crux is to match the natural-language input against the representations in the data sources or Web contents. The computer needs to understand ambiguous words and phrases such as "eternal city", "Ecstasy", "score", "player", and "metal band", and needs to match them against answer candidates such as "Ennio Morricone is a composer born in Rome, who became famous for the film music of several westerns. His Ecstasy of Gold was later performed also by cellist Yo-Yo Ma and by Metallica." For the matchmaking, the computer needs to solve difficult problems of *named entity disambiguation (NED)* (e.g., "eternal city" means Rome) and *word sense disambiguation (WSD)* (e.g., "cellist" matches "string player").

Example 2: Consider an entity search engine or a text analytics tool that makes recommendations for *related entities*. For example, when the user has discovered Ennio Morricone, the system may point out: "You may also be interested in Sergio Leone." A good system should augment this recommendation with an *explanation* of how Leone and Morricone are related. For example, the system could state that both are born in Rome, drawing from the facts in a KB or structured Web data. A more elaborated hint could be that Leone directed the dollar trilogy and Morricone composed the music for these movies, or more simply that both contributed to the movie "The Good, the Bad, and the Ugly". Here, both "born in" and "contributed to" are binary relations, where the former may be explicit in a KB and the latter is a priori merely a verbal phrase in a large text corpus. For being able to generate such informative explanations, the computer needs to have large dictionaries of relational paraphrases and understand how their synonyms and other lexical properties.

In this paper, we give an overview of our recent and ongoing work along these lines. We demonstrate a virtuous cycle in harvesting knowledge from large data and text collections and leveraging this knowledge in order to improve the annotation and interpretation of language in Web pages and social media. Specifically, we show how to build large dictionaries of names and paraphrases for entities and relations, and how these help to disambiguate entity mentions in texts. So we move from natural language to explicit knowledge structures, and then apply this knowledge for better machine-reading of language.

Section 2 reviews the transition from traditional CL resources to Web-scale KBs. Section 3 shows that KBs are key to building comprehensive high-quality dictionaries that connect names and phrases with entities and relationships. This in turn is a major asset for disambiguating surface names of entities in natural-language texts like news or social media, as explained in Section 4. Finally, Section 5 discusses the use case of how all this is beneficial for improving question answering over KBs and Linked Data sources.

2 Background on Knowledge Bases

The most widely used, traditional-style CL resource is the WordNet thesaurus [12]: a lexical collection of *words* and their word *senses*. Each word, such as "player", is mapped to one or more (usually more) concepts, and each concept is represented by its synonymous words that express the concept: a so-called synset accompanied by a short gloss. Two examples for synsets (with glosses in parentheses) are $\{player, participant\}$ (*a person who participates in or is skilled at some game*) and $\{musician, instrumentalist, player\}$ (*someone who plays a musical instrument as a profession*). These concepts are organized in a DAG-like hierarchy, with generalizations (hypernyms) such as $contestant$ (which, for example, has another child $athlete$) for the first sense of "player" and $performing\ artist$ for the second sense, as well as specializations (hyponyms) such as $football\ player$ or $organist$, $singer$, $soloist$, etc. WordNet contains more than 100,000 concepts and more than 200,000 word-sense pairs, all hand-crafted. It does not rigorously distinguish between classes that have entities as instances, e.g., $football\ player$, and general concepts, e.g., $harmony$, $sunset$, etc. Nevertheless, the class hierarchy of WordNet is the world's most comprehensive taxonomy of entity types.

The main deficiency of WordNet is that its classes have few instances; for example, WordNet does not know any football player or organist, and merely a few dozen singers. KB projects like YAGO closed this gap by 1) harvesting individual entities from Wikipedia and similar sources (e.g., geonames.org or musicbrainz.org), and 2) automatially mapping these entities into their proper WordNet classes. To this end, YAGO uses a noun phrase parser (a CL technique) to analyze the names of Wikipedia categories and identify their head words (e.g., "composers" in "Italian composers of film scores"), which determine candidates for superclasses of a given category. This often leaves ambiguity (e.g., "score" in the sense of grading someone's performance or in the sense of a musical composition) and also nonsensical candidates (e.g., "member" – of what?); YAGO uses simple but powerful heuristics for disambiguation. In total, this procedure yields ca. 10 million entities ontologically organized into ca. 350,000 classes: all WordNet classes plus those from Wikipedia that can be successfully mapped. Manual assessment over samples (with statistical confidence) shows that the mappings are correct in 95% of all cases. Together with relational facts derived from Wikipedia infoboxes and all provenance and context data (extraction rules or patterns, extraction source, etc.), YAGO contains nearly half a billion RDF triples. Interestingly, the entire construction of YAGO takes only a few hours on a reasonably configured server – there is no need for Map-Reduce parallelism or other kinds of scale-out techniques.

A number of projects have taken this line of automatic KB construction further in various ways:

1) increasing the number of facts about entities, by more aggressively tapping into infoboxes and other structured sources – notable examples are DBpedia [3] and freebase.com;
2) extending the KB with multilingual names of entities and classes – examples are UWN [8, 9], BabelNet [37], and WikiNet [35], and creating cross-lingual mappings between entity infoboxes [38];
3) collecting more entities from the Web, covering the long tail of out-of-Wikipedia entities – see, for example, the work on WebSets [7] and on instances-and-attributes discovery [1, 41];
4) collecting more fine-grained classes from the Web and placing them in the class taxonomy – Probase [54] and the work on doubly anchored patterns [22] are examples;
5) discovering and compiling new relations and their instances in an "Open Information Extraction" manner – from natural-language texts (e.g., TextRunner/ReVerb [4, 10]), from Web tables and lists (e.g., [25, 27, 49, 51]), by ab-initio machine learning (e.g., NELL [5]) or by crowdsourcing (e.g., ConceptNet [17, 45]).

3 From Language to Knowledge: Paraphrases at Web Scale

Names and keyphrases of entities: KBs are not an end by themselves; they allow us to connect Web contents with entities and their semantic properties, and thus enable more intelligent interpretation of contents. Therefore,

it is crucial that a KB also captures *surface names* for entities, as a counterpart to the WordNet synsets for general concepts. For example, for people, we should compile official full names (e.g., "Diana Frances Spencer") as well as short names (e.g., "Diana Spencer", "Lady Diana"), nicknames (e.g., "Lady Di"), role names (e.g., "Princess of Wales", "first wife of Prince Charles"), and other paraphrases (e.g., "queen of hearts"). For locations, organizations, and products, an even wider variety of equivalent names often exists. In addition to such names and paraphrases, it is often useful to know further *keyphrases* that are strongly connected with entities. For example, "British Royal Family" or "Paris car crash" with Lady Diana. Such keyphrases can be mined from large corpora by identifying noun phrases (via part-of-speech tagging and chunk parsing, both CL techniques) and weighing them by mutual information (MI, aka. relative entropy) with the name of the given entity [20, 48]. We have systematically compiled both names and keyphrases from Wikipedia redirects, href anchors, headings, and references, thus populating the two YAGO relations $means(name, entity)$ and $hasKeyphrase(entity, phrase)$ with more than 10 million and 80 million triples, respectively. At such input scales (the English Wikipedia contains more than 100 million href links) and output sizes, Map-Reduce techniques come in handy for gathering candidates and computing MI weights of phrases. Recently, Google has compiled a similar and even larger dictionary from their Web index, by considering href anchors pointing to Wikipedia articles [46].

Relational patterns: In high-quality KBs, the facts between entities are limited to a small set of prespecified relations. YAGO knows ca. 100 relations; DBpedia and Freebase provide a few thousand, but these are dominated by attribute-style properties with literals as arguments, e.g., hasRevenue, hasPopulation, hasGDP, hasCallingCode, etc. Interesting methods for compiling paraphrases for verb relations have been developed in [16, 23, 31, 26, 52]. Until recently, only ReVerb [10] offered a larger number of *relational patterns*. However, these are verbal phrases between noun phrases rather than properly typed relations, and consequently exhibit a large amount of noise. Examples of the resulting "factoid" triples are:

⟨ *"Carlos", "composed for", "voice and keyboards"* ⟩,
⟨ *"Maestro Morricone", "composed for", "the film"* ⟩,
⟨ *"Ennio Morricone", "was expected to win", "the Oscar"* ⟩,
⟨ *"Coulais", "has not won", "any of them"* ⟩.

Note that the first two patterns look identical but have different semantics, as the first refers to instruments as right-hand arguments and the second to movies. The other two patterns likely express the same relation (not winning an award), but this is not captured here as patterns are merely surface forms. Finally note that the arguments of the relational patterns are mere phrases, with ambiguous meanings (Carlos, the terrorist, or Juan Carlos, the Spanish king, or the composer Wendy Carlos or . . . ? Are Ennio Morricone and Maestro Morricone the same person?) and not necessarily denoting an entity at all ("any of them").

SOL patterns: This lack of semantic rigor motivated our approach, where we leverage the existing KB of canonicalized entities and their rich type system. We have compiled, by mining Web corpora, *relational paraphases* and organize them into *pattern synsets* with each synset having a *type signature*. To this end, we define a notion of syntactic-ontological-lexical patterns, *SOL patterns* for short, which are frequently occurring sequences of a) words (in lemmatized form, e.g., infinitive for verbs), b) part-of-speech tags (e.g., ADJ for adjectives or PRP$ for possessive pronouns), c) wildcards "∗" that stand for arbitrary words or word sequences, and d) semantic types as placeholders for arbitrary entities of a specific type (e.g., instruments, songs, movies, etc.). For example,

$\langle musician \rangle \ * \ compose \ for \ * \ \langle instrument \rangle$

is the SOL pattern that adds typing to the first example above, while a different SOL pattern

$\langle musician \rangle \ * \ compose \ for \ * \ \langle movie \rangle$

could be inferred from occurrences of individual musicians with individual movies and the "compose for" phrase. Further SOL patterns could be

$\langle person \rangle \ * \ honor \ by \ * \ \langle award \rangle$,
$\langle person \rangle \ * \ not \ win \ * \ \langle award \rangle$, or
$\langle person \rangle \ * \ disappointed \ * \ PRP\$ \ nomination \ * \ \langle award \rangle$,

the latter being derived from sentences such as "... was disappointed that her nomination did not result in ...". The second and the third pattern could be combined to form a *pattern synset* of equivalent patterns.

Big data methods: Our method for mining SOL patterns and organizing them into a pattern taxonomy proceeds in four steps: 1) extracting type-agnostic patterns, 2) enhancing these patterns with type signatures, 3) identifying synsets of equivalent SOL patterns, 4) inferring subsumptions among pattern synsets. For step 1 we employ techniques for *frequent sequence mining* [15, 14], after detecting entity names in sentences and mapping them to entities registered in a KB. Here we utilize the large dictionary names and their potential meanings, discussed in Section 2; the actual disambiguation method is discussed in Section 4. As we process many millions or even billions of sentences, the frequent sequence mining makes use of Map-Reduce parallelization (see [33] for details). For step 2 we replace entities by their semantic types and then compute frequent sequences with *type generalizations*. For example, the pattern

$\langle organist \rangle * composed\ for * \langle western\ movie \rangle$

may not be frequent enough, but then we lift the pattern into $\langle musician \rangle \ldots \langle movie \rangle$ or $\langle artist \rangle \ldots \langle movie \rangle$, etc. In steps 3 and 4 we consider the subsumption and equivalence of the resulting SOL patterns. To this end, we compare the *support sets* of patterns, i.e., the sets of entity pairs that co-occur with patterns. We say that a pattern p, say

$\langle singer \rangle * PRP\$\ ADJ\ voice\ in * \langle song \rangle$,

is subsumed by pattern q, e.g.,

$\langle musician \rangle * performed * \langle song \rangle$,

if the type signature of q is more general than (or the same as that of) p and the support set of p is, to a large extent, contained in the support set of q. The degree of set containment and the confidence in the pattern subsumption are quantified by statistical measures. Mutual subsumption between two patterns then yields synsets of (approximately) equivalent patterns. Finally, step 4 post-processes the output to ensure that the pattern taxonomy forms a proper DAG without cycles. Steps 3 and 4 operate on a prefix tree that encodes patterns and their support sets. All steps are parallelized using the Map-Reduce platform Hadoop.

Further details of this method are described in [33]. The resulting pattern taxonomy is called PATTY, and is available at the Web site `http://www.mpi-inf.mpg.de/yago-naga/patty/`. We currently offer PATTY collections built from the ClueWeb'09 corpus (a large crawl with 500 million English Web pages) and from the full text of the English Wikipedia (ca. 4 million articles). The former is larger; the latter is cleaner and contains more than 300,000 relational patterns with a sampling-based accuracy of 85% determined by human judges. Our Web site includes demos for several tasks [34]:

1) PATTY as a thesaurus of relational synsets including paraphrases for the canonicalized relations offered by DBpedia and YAGO,
2) schema-free search over entities and relational patterns, treating PATTY as a database of RDF triples, and
3) PATTY as a tool for explaining the relatedness of given entities (cf. [11, 29]).

4 From Knowledge to Language: Named Entity Disambiguation

NED problem: To demonstrate the enormous benefits of the compiled KB and dictionaries of names, keyphrases, and patterns, we now turn to the problem of named entity disambiguation, *NED* for short. Consider the example text "Carlos played the Moog in the Ludwig van scenes and arranged an electronic chorus for the fourth movement of the Ninth". There are standard techniques from computational linguistics for detecting spans of words that potentially denote named entities, so-called *mentions* of entities. Here, "Carlos", "Moog", "Ludwig van", and "the Ninth" are mentions that can be detected by shallow parsing and noun-phrase analysis, using conditional random fields [13]. This step also faces ambiguity, e.g., considering "Ludwig van" vs. "Ludwig van scenes" as candidate mentions, but the methodology is fairly mature. The task that NED subsequently needs to solve is to map each of the four mentions to exactly one (or, alternatively, at most one) entity registered in a KB.

The candidate space for the mapping is often huge. For example, Wikipedia and thus YAGO know four people with last name "Carlos" and more than 50 with first name "Carlos", including a king, a terrorist, many athletes, many musicians, etc. In addition, there are also cities, movies, fictional characters, and a guitar brand named "Carlos". There are more than 20 ninth symphonies, and "Ninth" has all kinds of other meanings as well (e.g., the Ninth Avenue in Manhattan).

The NED problem is tackled by combining different ingredients from the following considerations [6, 30, 24]. Our method combines all of them in a judicious manner [19].

- **Popularity of entities:** An ambiguous name is more likely to denote a prominent entity than some lesser known person or location in the long tail. For example, "Moog" is more likely to mean the synthesizer or its inventor Robert Moog, rather than the painter and poet Peter Moog or the Moog software package. The techniques for mining name-entity pairs (see Section 3) can easily collect frequency information from href anchors or could consider additional information like the lengths and link degrees of Wikipedia articles, in order to estimate the conditional probability $P[e|n]$ for entity e when observing name n. This notion of popularity is good as a *prior* for NED, but always needs to be combined with other components.
- **Similarity of contexts:** Mentions are surrounded by text, and this context can be compared to textual descriptions of candidate entities. One way of doing this is to construct bags-of-words or statistical language models (using IR techniques) over words, bigrams (word pairs), or entire phrases in the mention's proximity of the input text, on one hand, and over the same style of tokens in the Wikipedia article of an entity. Once these models are built, similarity measures like cosine (for bags-of-words vectors) or Kullback-Leibler divergence (for language models) can be used to assess how well a candidate entity e fits with a mention m, given the surrounding contexts $cxt(\ldots)$: $P[e|n, cxt(e), cxt(n)] \sim sim(cxt(e), cxt(n))$.
- **Coherence among entities:** Entities occur together in a text not at random, but only if they are semantically related to each other. For example, why would Carlos, the terrorist, be mentioned with the Moog synthesizer in a news article or blog posting? Instead, it is more likely that a musician like Wendy Carlos co-occurs with instruments like the Moog, regardless of which surface forms are used to refer to these entities. This suggests a notion of coherence for a set of entities. When entities are registered in a KB and have corresponding Wikipedia articles, one way of quantifying the coherence between two entities is based on overlap measures between the articles' incoming links [30]. For tractability, the coherence of a set E with more than two entities is factorized into pair-wise coherence: $coh(E) = P[\{e|e \in E\}] = \prod_{a,b \in E} P[a, b] \sim overlap(in(a), in(b))$.

Keyphrases for similarity: In our work on AIDA, we developed a particularly powerful kind of context-similarity model, based on the keyphrases of entities gathered by the methods of Section 3. Instead of using all words or bigrams in comparing the context of a mention and the context of an entity, we use only entity-specific keyphrases. For example, Wendy Carlos has associated keyphrases "Moog synthesizer", "Well Tempered Synthesizer", "electronic musician", etc. These are partly matched in the input text surrounding the mention "Carlos" (our example above). In [48], we developed a sophisticated model for *partial keyphrase matching* which we adopted and extended to the task of NED. This model considers the proximity of words in a partial match, uses mutual-information weights for both words and phrases, and aggregates scores over all keyphrases of a given entity. For example, when comparing Beethoven's keyphrase "Ninth Symphony's Ode to Joy" could be against the sentence "...Ode in the fourth movement of the Ninth ...", the score contribution is proportional to the total weight of the two matched words and inversely proportional to the length of the text window that contains them. This is a very powerful model, but one needs a lot of care for an efficient implementation, using big-data techniques like min-hash sketches and locality-sensitive hashing [43].

Graph algorithms for coherence: AIDA expresses the joint inference on mention-entity context similarity and entity-entity coherence as a problem of computing dense subgraphs. It builds a graph with mentions and candidate entities as nodes. A combination of keyphrase-based similarity and popularity scores is used for weights of mention-entity edges; entity-entity edges are weighed by link-overlap-based coherence. Figure 4

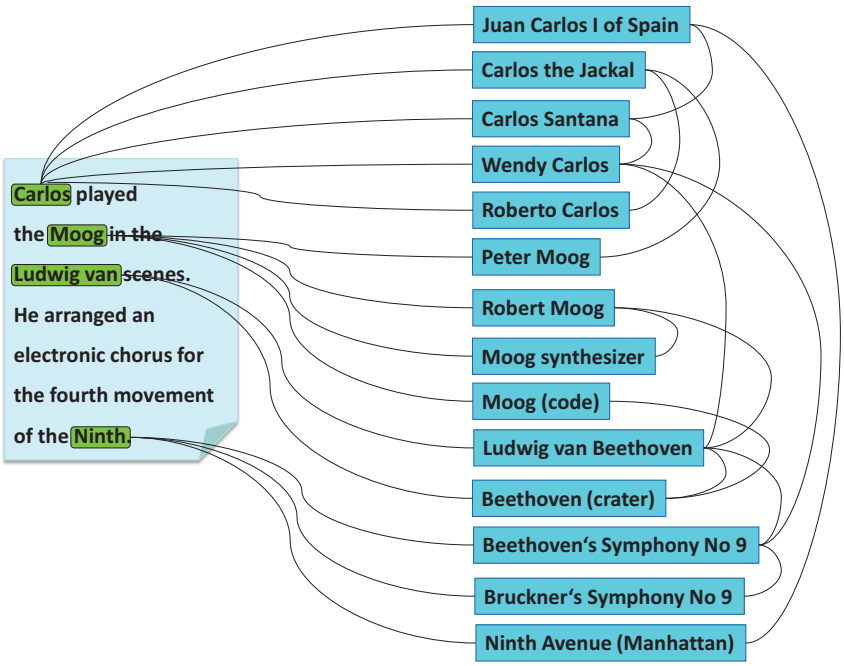


Figure 1: Example graph for named entity disambiguation

shows an excerpt for our running example. Realistic graphs can easily have many thousands of nodes, say for a news article that uses common last names or highly ambiguous phrases.

Given such a graph, the task of NED becomes a problem of computing a dense subgraph with high edge weights. Inspired by work on social network analysis [44], that is, "big data", we actually pursue the goal of finding a subgraph whose minimum weighted degree (the total weight of a node's incident edges) is as large as possible, with appropriate constraints for the NED mapping. The intuition here is that NED should be done such that the weakest link is as strong as possible. Not surprisingly, this problem is NP-hard; we devised practically viable approximation algorithms [19]. The NED quality of this method achieves top results on benchmark tasks of the CL community. AIDA is available online at `http://www.mpi-inf.mpg.de/yago-naga/aida/`.

Big data methods: In this CL-centered work, we leveraged large-scale KB's and dictionaries as key assets as well as a variety of methods typical for big data analytics: dictionaries and statistics from Web data, approximate matching, dense-subgraph computation. One of the next steps that we aim for is to apply NED to an entire corpus in a near-real-time and high-throughput manner. As a use case, consider that an analyst wants to study the media coverage of a country's politicians in one day's news (in thousands of newspapers) and social media, and also investigate the relationships with other people and organizations as expressed in this day's media. Another scenario is that a journalist wants to track a politician or celebrity over an extended time period in a large media archive, to discover patterns and trends in the entities and contexts. In both of these use cases, we need to scale up NED to process huge amounts of input documents. While this can be scaled out by partitioning the inputs, there may be better choices (e.g., in terms of memory consumption and parallel throughput) like partitioning the dictionary or dedicating compute nodes to specific subsets of target entities.

5 Use Case: Translating Questions into Queries

To illustrate how the knowledge-language interplay helps for advanced tasks, reconsider our initial question answering scenario where a user asks: "Which composer from the eternal city wrote the score for the Ecstasy

scene?" We have developed methods and a prototype system, called DEANNA [50], for automatically translating the natural language question into a structured SPARQL query that can be evaluated over subject-predicate-object (SPO) data in the RDF model. The choice of RDF and SPARQL is motivated by their schema-relaxed or schema-free nature and the fact that the Web of Linked Data provides huge amounts of informative and diverse SPO triples. The example question is translated into the query:

Select ?p Where {
?p type composer . ?p bornIn Rome . ?p created Ecstasy_of_Gold . }

where *?p* is a variable and the appearance of the same variable in different triple patterns denotes joins.

A major difficulty in this question-to-query translation lies in mapping phrases like "composer", "eternal city", or "Ecstasy" into classes and entities and phrases like "from", and "score for" into relations of the underlying KBs and other RDF sources. This resembles the NED problem discussed in Section 4, but we face a more general disambiguation problem here. A priori, a word like "score" could be either a class (e.g., soundtracks), or an entity (e.g., the movie "The Score", or a music album, or a company), or even a relation (e.g., hasSoundtrack between movies and music pieces, or scoresFor between football players and their clubs). Our DEANNA system generates all these potential meanings, using the dictionaries and pattern taxonomies discussed in Section 3. DEANNA then imposes type constraints on the feasible combinations. If we consider mapping a phrase like "score for" into the relation wroteMusic, we constrain the left and right arguments of the phrase, "Which composer" and "Ecstasy", to be of types composer and piece of music, respectively. These and other constraints are encoded into an integer linear program (ILP) for a combined selection-and-assignment problem: select appropriate phrases from the input question and assign them to classes, entities, or relations. The objective function is to maximize the coherence among the chosen candidates. The resulting ILP is not exactly small, even for short inputs, but modern solvers (e.g., `http://www.gurobi.com`) can handle this in reasonable time, usually a few seconds.

Further details on this method are given in [50]. For the example question, DEANNA generates the SPARQL query shown above, and evaluating this query over YAGO returns the correct answer: Ennio Morricone.

6 Conclusion

This paper has given an overview of our work on tackling various problems in computational linguistics, by methods that are typical for big data analytics: frequent sequence mining at Web scale, co-occurrence statistics, approximate matching, graph algorithms, scalable optimization such as ILP. Details on our work can be found in the original papers cited in each section.

We believe that these methods and the semantic assets from large knowledge bases and the Web of Linked Data are game-changers for some notoriously hard problems in computational linguistics. Text understanding in terms of entities and relationships, and question answering in natural language can greatly benefit from big data and the accompanying methodology.

References

[1] Enrique Alfonseca, Marius Pasca, Enrique Robledo-Arnuncio: Acquisition of Instance Attributes via Labeled and Related Instances. SIGIR 2010: 58-65

[2] Ion Androutsopoulos, Prodromos Malakasiotis: A Survey of Paraphrasing and Textual Entailment Methods, Journal of Artificial Intelligence Research 38: 135–187, 2010

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary G. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC/ASWC 2007: 722-735

[4] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open Information Extraction from the Web. IJCAI 2007: 2670-2676

[5] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., Tom M. Mitchell: Toward an Architecture for Never-Ending Language Learning. AAAI 2010: 1306-1313

[6] Silviu Cucerzan: Large-Scale Named Entity Disambiguation Based on Wikipedia Data. EMNLP-CoNLL 2007: 708-716

[7] Bhavana Bharat Dalvi, William W. Cohen, Jamie Callan: WebSets: Extracting Sets of Entities from the Web using Unsupervised Information Extraction. WSDM 2012: 243-252

[8] Gerard de Melo, Gerhard Weikum: MENTA: Inducing Multilingual Taxonomies from Wikipedia, CIKM 2010: 1099-1108

[9] Gerard de Melo, Gerhard Weikum: UWN: A Large Multilingual Lexical Knowledge Base, ACL (System Demonstrations) 2012: 151-156

[10] Anthony Fader, Stephen Soderland, Oren Etzioni: Identifying Relations for Open Information Extraction. EMNLP 2011: 1535-1545

[11] Lujun Fang, Anish Das Sarma, Cong Yu, Philip Bohannon: REX: Explaining Relationships between Entity Pairs. PVLDB 5(3): 241-252, 2011

[12] Christiane Fellbaum, George Miller (Editors): WordNet: An Electronic Lexical Database, MIT Press, 1998

[13] Jenny Rose Finkel, Trond Grenager, Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. ACL 2005: 363-370

[14] Gösta Grahne, Jianfei Zhu: Fast Algorithms for Frequent Itemset Mining Using FP-Trees. IEEE Transactions on Knowledge and Data Engineering 17(10): 1347-1362, 2005

[15] Jiawei Han, Jian Pei, Yiwen Yin, Runying Mao: Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery 8(1): 53-87, 2004

[16] Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Stijn De Saeger, Masaki Murata, Jun'ichi Kazama: Large-Scale Verb Entailment Acquisition from the Web. EMNLP 2009: 1172-1181

[17] Catherine Havasi, Robert Speer, and Jason Alonso. ConceptNet: A Lexical Resource for Common Sense Knowledge. In: Recent Advances in Natural Language Processing, Volume 5. John Benjamins Publ., 2009

[18] Tom Heath, Christian Bizer: Linked Data: Evolving the Web into a Global Data Space Morgan & Claypool Publishers 2011

[19] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, Gerhard Weikum: Robust Disambiguation of Named Entities in Text, EMNLP 2011: 782-792

[20] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Gerhard Weikum: YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia, Artificial Intelligence 2012

[21] IBM Journal of Research and Development 56(3-4), Special Issue on "This is Watson", 2012

[22] Zornitsa Kozareva, Ellen Riloff, Eduard H. Hovy: Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. ACL 2008: 1048-1056

[23] Zornitsa Kozareva, Eduard H. Hovy: Learning Arguments and Supertypes of Semantic Relations Using Recursive Patterns. ACL 2010: 1482-1491

[24] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, Soumen Chakrabarti: Collective Annotation of Wikipedia Entities in Web Text. KDD 2009: 457-466

[25] Girija Limaye, Sunita Sarawagi, Soumen Chakrabarti: Annotating and Searching Web Tables Using Entities, Types and Relationships. PVLDB 3(1): 1338-1347, 2010

[26] Dekang Lin, Patrick Pantel: DIRT – Discovery of Inference Rules from Text. KDD 2001: 323-328

[27] Ashwin Machanavajjhala, Arun Shankar Iyer, Philip Bohannon, Srujana Merugu: Collective Extraction from Heterogeneous Web Lists. WSDM 2011: 445-454

[28] Inderjeet Mani: Automatic Summarization, John Benjamin Publ., 2001

[29] Steffen Metzger, Shady Elbassuoni, Katja Hose, Ralf Schenkel: S3K: Seeking Statement-Supporting Top-K Witnesses. CIKM 2011: 37-46

[30] David N. Milne, Ian H. Witten: Learning to Link with Wikipedia. CIKM 2008: 509-518

[31] Thahir Mohamed, Estevam R. Hruschka Jr., Tom M. Mitchell: Discovering Relations between Noun Categories. EMNLP 2011: 1447-1455

[32] Ndapandula Nakashole, Martin Theobald, Gerhard Weikum: Scalable Knowledge Harvesting with High Precision and High Recall, WSDM 2011: 227-236

[33] Ndapandula Nakashole, Gerhard Weikum, Fabian Suchanek: PATTY: A Taxonomy of Relational Patterns with Semantic Types, EMNLP-CoNLL 2012: 1135-1145

[34] Ndapandula Nakashole, Gerhard Weikum, Fabian Suchanek: Discovering and Exploring Relations on the Web, PVLDB 5(12): 1982-1985, 2012

[35] Vivi Nastase, Michael Strube, Benjamin Boerschinger, Ccilia Zirn, Anas Elghafari: WikiNet: A Very Large Scale Multi-Lingual Concept Network. LREC 2010: 1015-1022

[36] Roberto Navigli: Word Sense Disambiguation: A survey. ACM Computing Surveys 41(2): 10:1-10:69, 2009

[37] Roberto Navigli, Simone Paolo Ponzetto: BabelNet: Building a Very Large Multilingual Semantic Network. ACL 2010: 216-225

[38] Thanh Hoang Nguyen, Viviane Moreira, Huong Nguyen, Hoa Nguyen, Juliana Freire: Multilingual Schema Matching for Wikipedia Infoboxes. PVLDB 5(2): 133-144, 2011

[39] Martha Palmer, Daniel Gildea, Nianwen Xue: Semantic Role Labeling, Morgan & Claypool Publishers, 2010

[40] Bo Pang, Lillian Lee: Opinion Mining and Sentiment Analysis, Foundations and Trends in Information Retrieval 2(1-2), Now Publishers, 2008

[41] Joseph Reisinger, Marius Pasca: Fine-Grained Class Label Markup of Search Queries. ACL 2011: 1200-1209

[42] Simone Paolo Ponzetto, Michael Strube: Deriving a Large-Scale Taxonomy from Wikipedia. AAAI 2007: 1440-1445

[43] Anand Rajaraman, Jeffrey David Ullman: Mining of Massive Datasets, Cambridge University Press, 2011

[44] Mauro Sozio, Aristides Gionis: The Community-Search Problem and How to Plan a Successful Cocktail Party. KDD 2010: 939-948

[45] Robert Speer, Catherine Havasi, and Harshit Surana Using Verbosity: Common Sense Data from Games with a Purpose. FLAIRS 2010: 104-109

[46] Valentin I. Spitkovsky, Angel X. Chang: A Cross-Lingual Dictionary for English Wikipedia Concepts, LREC 2012: 3168-3175

[47] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum: Yago: a Core of Semantic Knowledge. WWW 2007: 697-706

[48] Bilyana Taneva, Mouna Kacimi, Gerhard Weikum: Finding Images of Difficult Entities in the Long Tail. CIKM 2011: 189-194

[49] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, Chung Wu: Recovering Semantics of Tables on the Web. PVLDB 4(9): 528-538, 2011

[50] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, Gerhard Weikum: Natural Language Questions for the Web of Data, EMNLP-CoNLL 2012: 379-390

[51] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, Surajit Chaudhuri: InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. SIGMOD Conference 2012: 97-108

[52] Limin Yao, Aria Haghighi, Sebastian Riedel, Andrew McCallum: Structured Relation Discovery using Generative Models. EMNLP 2011: 1456-1466

[53] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, Gerhard Weikum: AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. PVLDB 4(12): 1450-1453, 2011

[54] Wentao Wu, Hongsong Li, Haixun Wang, Kenny Qili Zhu: Probase: a Probabilistic Taxonomy for Text Understanding. SIGMOD Conference 2012: 481-492

Bisque:Advances in Bioimage Databases

Kristian Kvilekval, Dmitry Fedorov, Utkarsh Gaur,
Steve Goff, Nirav Merchant, B.S. Manjunath, Ambuj Singh

Abstract

Biological image databases have quickly replaced the personal media collections of individual scientists. Such databases permit objective comparisons, benchmarking, and data-driven science. As these collections have grown using advanced (and automated) imaging tools and microscopes, scientists need high-throughput large-scale statistical analysis of the data.

Traditional databases and standalone analysis tools are not suited for image-based scientific endeavors due to subjectivity, non-uniformity and uncertainty of the primary data and their analyses. This paper describes our image-database platform Bisque, which combines flexible data structuring, uncertain data management and high-throughput analysis. In particular, we examine: (i) Management of scientific images and metadata for experimental science where the data model may change from experiment to experiment; (ii) Providing easy provisioning for high-throughput and large-scale image analysis using cluster/cloud resources; (iii) Strategies for managing uncertainty in measurement and analysis so that important aspects of the data are not prematurely filtered.

1 Challenges for Bioimage Researchers

Current research in biology is increasingly dependent on conceptual and quantitative approaches from information sciences, ranging from theory through models to computational tools [6]. Ready availability of new microscopes and imaging techniques has produced vast amounts of multi-dimensional images and metadata. The introduction of new models, measurements, and methods has produced a wealth of data using image-based evidence [24]. Two notable examples of image-based studies are cellular Alzheimer's studies and plant genetics.

In a recent Alzheimer's study, the ability to reliably detect nuclei in three dimensions was critical to quantitative analysis [25]. The use of nuclei detection also finds use in a wide range of applications such as the accurate determination of how an organism is perturbed by genetic mutations, treatment with drugs, or by injury. Additionally, nuclei centroid locations can be used for further analysis, such as cell membrane segmentation or to initialize and validate computational models of cells and their patterns.

In the plant domain, technologies for quantifying plant development are underdeveloped relative to technologies for studying and altering genomes. As a result, information about plant gene function inherent in mutant phenotypes or natural genetic variation remains hidden. For example, plant scientists are trying to uncover gene function by studying seedling growth and development using high throughput image analysis [19].

In both cases, researchers dependent on images as experimental evidence face the daunting task of managing, analyzing and sharing images in addition to gaining and providing access to analysis methods and results [1]. In

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

the following paragraphs, we enumerate several challenges commonly faced by scientific researchers working with large-scale image data.

Growth of data: automated imaging and large scale processing Image-based experiments can produce hundreds to millions of images per experiment. For example, automated image-based phenotyping [19] produces several terabytes of image data. In addition to the data management problem, researchers are increasingly dependent on automated or semi-automated image analysis [32] due to large amounts of images involved in modern biological studies. Researchers working with very large datasets must thus take advantage of scalable computational resources. The results of analyses also pose a data management problem requiring careful management of initial (raw) and processed data while maintaining relationships to analyzed results.

Dealing with novelty and reproducibility in scientific experiments Biological image data models require a flexibility not usually needed by traditional database applications. In fact, the key to properly interpreting biological images is the experimental and image related metadata captured during experimental preparation and imaging. For example, a sample's biology including genetic mutations, or imaging techniques such as antibody staining are not discernible from the available pixel data. Furthermore, image data is often multi-dimensional including volume and time as well as being multi-channel (i.e. antibody labels rendered as false color images). Biology labs employ diverse experimental procedures [36] and continually invent new procedures and preparations resulting in unique local workflows [28]. New measurements, analysis, and statistics have also become increasingly complex and challenging [8]. Novel analysis techniques and results may require changes to the underlying data model [33] in order to be made available along with original data. While several laboratory image database systems have been developed, database schema rigidity has often become problematic as requirements evolve due to changes in experimental protocols and required analyses.

As computational image analysis is further integrated into the scientific process, accurate tracking of experimental results has become a primary concern [20]. Maintenance of original data while ensuring accurate tracking of results is fast becoming a requirement. In order to ensure accurate provenance, result data needs to be reliably marked by a tamper-resistant system in which the analysis and the resultant data become fixed once added to the system. Provenance provides security, confidence and traceability for resulting analysis.

Summarizing, comparing, disseminating, re-evaluating, mining Image analysis workflows can create large amounts of data that must be summarized, and compared for human understanding. Object identification image-analyses (i.e., segmentation and classification) can produce millions of spatial annotations per image (e.g., cells in tissue sample, microtubules within a cell). Furthermore, dissemination of large datasets can in itself be challenging. While some funding agencies do require access to published data for open science requirements [20], many researchers also realize the importance of data sharing and collaboration that can lead to better data validation, increased knowledge discovery, and cross-domain collaborations [37]. Even though researchers are usually willing to share data once published, strict security must be in place for works in progress. Achieving the goals of strict security and ease of sharing has proved challenging, and in many cases sharing of data has suffered.

Sources and management of uncertainty In an extensive number of scientific investigations, there is a growing dependence on large volumes of image and video data, and subsequently their analysis. Moreover, most image-based measurements are inherently uncertain and when used in extensive workflows accumulate errors if uncertainty estimation is not properly propagated. While it is possible to record and propagate uncertainty of measurements throughout processing pipeline, most systems rely on some sort of thresholding of results at each processing step. Even if the uncertainty measurement is recorded, the data produced (containing inherent ambiguities and uncertainties) can pose significant challenges to information processing systems.

2 Bisque: a Scientific-Image Database for Image Analysis

Bisque is a novel image database and analysis system built for biologists working with digital artifacts (such as images and video) as a fundamental tool for gathering evidence. Bisque allows users to store, access, share and process biological images, files and other resources from any web-connected device. The system is built to manage and analyze both data artifacts and metadata for large scale datasets by utilizing emerging large-scale computing cloud computing principles. The Bisque system [17] has been in development for several years and is being used for large scale image management and analysis. We highlight some of the unique features and services provided by Bisque for scientific-image management and analysis.

2.1 Interpreting scientific images: pixels to understanding

Biological digital artifacts including microscopic images and sensor data are only interpretable with the appropriate context. For example, an RGB image of cells does not explain what tissue the cells are embedded in, what the sample came from, nor how it was treated. It is vitally important to preserve this contextual metadata if we are to have any hope of re-utilizing (through analysis or mining) scientific data. In some cases, the context is well understood or previously defined. For example, the human genome project required researchers to contribute identified sequences to a known corpus in well defined formats such that the results could be easily shared and understood by all [15]. The majority of scientific experiments are not carried out with such a degree of coordination between members of the scientific community. Nor it is desirable as the time and effort needed to make them compatible must be paid in advance before the results are known. Instead, we believe, it is better to capture the experimental context as the researchers describe it and provide tools to make it available and searchable later.

2.2 Context and metadata : flexible modeling for scientific context

At the core of a Bisque site is the Bisque data service allowing Bisque resources to be stored and queried. Resources allow flexible modeling of context and experimental data permitting experimenters and practitioners to model their metadata as their experiments require. Each resource is comprised of a metadata record (a nested tree of key-value pairs) with an optional binary component. Resources can be easily linked together to provide common context. For example, in Fig. 1) an image may be linked to both a project and sample resources allowing each to be described independently and completely. Bisque resources form graphs which are used for data modeling, and data retrieval. The system is schemaless in that scientists define the data model per lab, experiment etc.

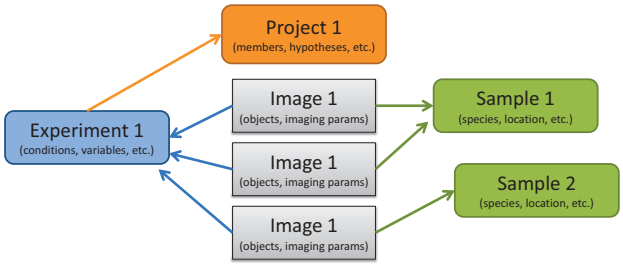


Figure 1: Bisque resources are document graphs linking contextual metadata. Here, images are linked to both a experiment and sample resources allowing each to be described independently, while an experiment is linked to a project. These graphs are also used for data retrieval. Annotations in each of the nodes are schemaless in that scientists are free to define complex trees of name-value-type tuples on the fly.

Resources are easily represented by linked XML documents. Every Bisque resource has a unique URL and may be accessed using common web protocols. The Bisque data service permits queries of top level resources based on elements of the Bisque resource graph.

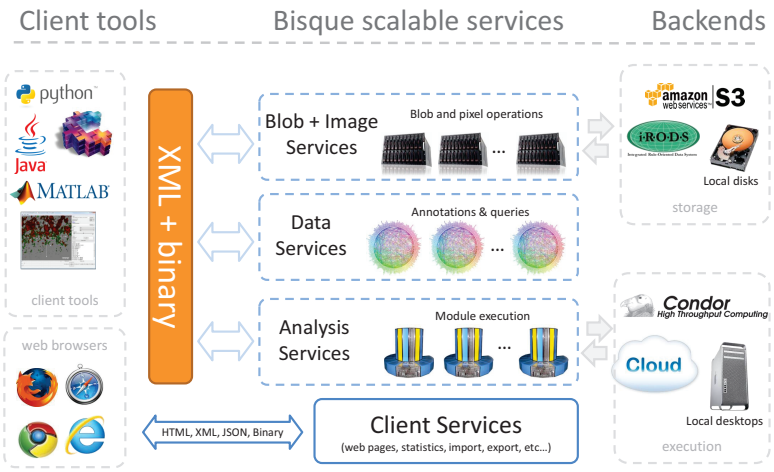


Figure 2: Bisque is implemented as a scalable and modular web-service architecture. *Image Servers* store and manipulate images. Data servers provide flexible metadata storage. Execution servers house executable modules. The client-server architecture seamlessly integrates services across storage and processing hardware. Communication between various components is performed in a RESTful manner through HTTP requests carrying XML and JSON. Web clients use dynamic web components built with AJAX. Backends seamlessly provide access to local resources as well as large scalable resources like Amazon S3 or iRODS for data storage or Condor grids for computation.

2.3 Large-scale storage and processing

The Bisque system has been designed for scalability from top to bottom. When installed, a Bisque site (Fig. 2) is comprised of many micro-services, each performing a specialized task. The majority of Bisque services simply manipulate and interpret the resource metadata. Other specialized micro-servers work with particular binary resources. For example, the Bisque image server is capable of performing common image operations (channel mapping, slicing, format conversion, etc.) on many image formats (both proprietary and open).

Scalability requires that the system can grow to support very large datasets, large numbers of users, and large analysis workflows. User scalability is provided by the fact that Bisque servers may be replicated and may be scaled using well-understood web-server scaling techniques (reverse proxies, replicated servers, etc). Furthermore, a Bisque site can take advantage of computational scaling for image analysis using cluster and grid resources in order to process very large datasets. Two internal components directly take advantage of these services: the blob server for large-scale storage and the module engine for parallel and scalable analysis. The blob service allows binary objects to be stored on a variety of storage systems from local file-systems to large distributed file-stores such as iRods [16], HDFS [14], or Amazon S3 [3]. Users are freed from storage issues and can access their resource from any Internet-accessible site. The module engine allows developers to integrate analysis algorithms with one or more Bisque sites. The engine performs both Bisque interfacing and job management for analysis routines. The engine can also be configured to use local resource or to use external job schedulers such as Condor [35] permitting access to cluster and grid resources. A key feature is that developers can often develop locally and deploy on grid resources with no changes to the analysis code. This is due to the fact that analysis modules using Bisque services need only make HTTP accesses.

2.4 Extensibility and availability

Bisque is extensible through both service development and analysis development. Example extended services include the statistics and summarizing service, and a search-by-content service. Bisque services are RESTful [22] allowing access from any programming environment that can make HTTP request.

Bisque views analysis modules simply as specialized websites (providing specific http endpoints) offering

analysis services. Many example analysis modules are included with Bisque and can be used as a guide for development. User modules can then be linked to an existing Bisque site (see Fig. 3). Analysis developers can install a small package which wraps common developed code to provide the needed protocol elements.

2.5 Provenance and trust

A core feature of Bisque system is that resources retain their provenance. When a resource is created or modified, it is marked by the analysis or session of the action. The Module Execution Record (MEX) can be linked to form the complete provenance of any item in the database from original upload or creation to the resulting statistics from the analysis of a dataset. Since provenance maintenance is managed by the system itself, users (and reviewers) can be assured that items are tamper-free, thus providing a level of trust in the data. Users can user provenance data to follow faulty data or simply manage chains of analysis.

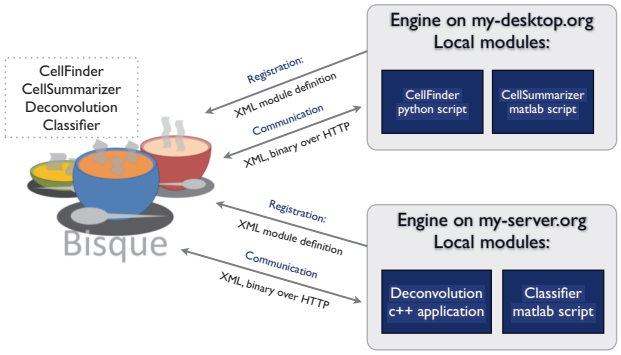


Figure 3: Bisque image analysis capabilities can be augmented by adding new services available on external computational resources. These external resources can be arbitrarily complex and harness available local power seamlessly from the main system. A Bisque module is simply a web-enabled internet end-point which adheres to a specific Bisque API for communication. The API is based on simple XML and HTTP/HTTPS transfers. Each end point can be made available by running a Bisque Engine Server that wraps all the communication and can enable simple Matlab or Python scripts to become Bisque modules. Each module is registered to the system by providing an XML description that includes required inputs and outputs.

3 Scientific Data Services

Scientific labs can often produce more data and analytical problems than they can handle. Each lab can search for or buy computational resources but these tasks can incur high overheads both in terms of people, money and time. With the advent of virtualized commodity computing, labs have new choices: utilizing free computational resources or renting large-scale resources.

Bisque users and developers have many choices on how to use the system: use a public installation such as the ones at Center for Bioimage Informatics [9] or iPlant [13], install a private instance on local hardware, or install an instance on a public cloud.

3.1 Available infrastructure and Cloud computing for e-sciences

The iPlant Collaborative [13] is an NSF-funded cyberinfrastructure (CI) effort directed towards the plant sciences community. The inherent interdisciplinary nature of plant sciences research produces diverse and complex data products that range from molecular sequences to satellite imagery as part of the discovery life cycle. With the constant creation of novel analysis algorithms, the advent and spread of large data repositories, and the need for collaborative data analysis, marshaling resources to effectively utilize these capabilities necessitates a highly flexible and scalable approach for implementing the underlying CI.

The iPlant infrastructure simultaneously supports multiple interdisciplinary projects providing essential features found in traditional science gateways as well as highly customized direct access to its underlying frameworks through use of APIs (Application Programming Interfaces). This allows the community to develop de novo applications. This approach allows us to serve broad community needs while providing flexible, secure, and creative utilization of our platform that is based on best practices and that leverages established computational resources.

Bisque is available for use by qualified plant science projects utilizing iPlant resources including the Data Store and High Performance Computing Infrastructure.

3.2 Large-scale processing using temporary resources

Commoditized computing services have become available from commercial providers such as Amazon, Google, Rackspace, and RightScale. In many cases, these are provided at low cost or freely for qualified projects. Bisque has been designed to work with such services allowing large-scale processing to be available to small labs.

For example, the Bisque project provides templates to deploy a Bisque server with a parallel Condor grid on Amazon EC2/S3. A scientist faced with analyzing a very large image dataset can, without much effort, create a "temporary" cluster-system for processing the data. Scientists can utilize these on-demand services to process very large datasets without the need to build large-infrastructure.

4 Future: Coping with Uncertainty

Imaging is at the core of many scientific discoveries, at scales varying from nano to astronomical observations. The information is captured in terms of raw pixel intensities and in multiple channels for color or hyperspectral imagery. These pixel values by themselves have very little meaning in most cases, and it is their spatial and temporal variations that are of much interest. Thus, most image analysis methods implicitly address the *uncertainty* in the observed values, whether it is for edge detection or segmentation.

The Bisque system is being extended to support *uncertainty* for confidence measures and spatial annotations which results from measurements and analysis. Two key challenges being addressed are: modeling uncertainty and computing with uncertain data.

4.1 Modeling uncertainty

Uncertainty in database systems can be modeled at the level of tuples or attributes. In the former case, the existence of a tuple is uncertain and a probability value is assigned to each tuple representing the confidence with which that particular tuple is present in the database. In the case of attribute uncertainty, there is certainty about the existence of the tuple but uncertainty about its specific attributes. Combinations of tuple and attribute uncertainties are also possible. Furthermore, dependence between attributes and tuples in a given relation or across relations can be modeled by probabilistic graphical models [29] in which random variables represent tuples or attributes. However, inference (the basis for answering queries) is difficult due to NP-completeness, and various approximations need to be employed.

Existing database literature in the area of uncertainty can be broadly divided into two categories: models and semantics, and query processing and index structures. The usually accepted model for evaluation of probabilistic queries is the "possible-worlds" model [4, 12, 18, 11] in which a set of possible certain worlds is defined for the database objects, the query is evaluated in these certain worlds, and the result is aggregated to return the answer. In general, such evaluation is computationally challenging and makes a straightforward implementation all but infeasible for large databases. A number of strategies have been developed to find subclasses of queries that can be answered efficiently. This includes the isolation of safe query plans [11, 4]. Other approaches combine uncertainty with lineage [5] and consider the decomposition of possible worlds into a manageable set of relations [2]. Aspects of completeness under various models of uncertainty have been considered [27].

Bisque is being extended to work with confidence tags for existential annotations and probability masks to map spatial uncertainty. We are also investigating vertex models for spatial uncertainty including uncertain points, lines and polygons. Initially, probabilistic data will be available to analysis methods and visualization systems in order to allow researchers to gain experience utilizing probabilistic data. For example, annotation

visualization will allow filtering results based on confidence and permit visualizing zones of uncertainty for spatial objects.

4.2 Computing and searching

Answering queries on probability density functions (*pdf*) has been well studied [7, 10, 34] under the Gaussian and Uniform *pdf*s. These assumptions allow for interesting and efficient index structures, and can be appropriate for the uncertainty of many of the individual measurements. They are too restrictive for *pdf*s that occur as a result of summarization, however, the observations being summarized may be generated by different mechanisms. For instance, a summary of the density of bipolar cells in a detached cat retina will have two peaks, corresponding to parts of the retina that are injured and healthy, respectively. Fitting a model distribution, such as a Gaussian, to the data is only appropriate when the data are well understood, and in a scientific database, the most interesting data to query are precisely the ones that are not well understood. Others have considered the indexing of uncertain categorical data [30], the use of Monte Carlo simulations and state space search to answer top-k queries [26, 31], and skyline queries [23]. The TRIO system [38] supports data uncertainty at different levels and lineage. The ORION project [21] is a recent work aimed at developing an advanced database system with direct support for uncertain data.

Analysis methods such as segmentation result in different shapes and sizes based on the method or the initial conditions and subsequent data mining (classification, proximity analysis) can produce different outcomes. Since the possible worlds are too numerous to examine or visualize, one challenge is to produce a set of "diverse" outcomes by sampling from the underlying segmentation uncertainty and providing them to a user or another analysis tool. Another interesting possibility is to extract features of a given image and use these to suggest an analysis method (e.g., segmentation, initial seed) that is likely to work well based on past experience.

We are currently working to extend Bisque with spatial join and filtering including probabilistic objects with spatial uncertainty. Initially we will focus on generating spatial indexes for probabilistic objects and visualization tools. These tools will allow researchers to filter for spatial properties. For example, a researcher might be interested in determining characteristics of labeled astrocyte cells lying "near" a blood vessel in the retina. Determining the extent of astrocyte is ambiguous due to dendrite extensions connecting cells. The uncertainty of the shape of final astrocyte cell needs to be factored into the spatial join at the last possible moment.

5 Conclusion

Scientists working with images as fundamental evidence face many challenges. Gathering, documenting, organizing, analyzing and disseminating original data are at the core scientific of the process. Large scale imaging presents issues at each of the activities. We have outlined the challenges that we have discovered while working with collaborators from the biological sciences and have outlined our system Bisque to assist researchers working with image data. Key amongst those challenges are: accurate collection and organization of metadata, and analysis and dissemination of original images, analysis results, and provenance. We have also outlined our efforts to support uncertain data within the Bisque system allowing researchers to manage the uncertainty of results throughout a scientific workflow.

References

[1] P. Andrews, I. Harper, and J. Swedlow. To 5d and beyond: Quantitative fluorescence microscopy in the postgenomic era. *Traffic*, 3(1):29–36, 2002.

[2] L. Antova, C. Koch, and D. Olteanu. 10106worlds and beyond: Efficient representation and processing of incomplete information. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 606 –615, april 2007.

[3] Amazon EC2 and S3. http://aws.amazon.com/.

[4] D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Tansaction on Knowledge Engineering*, 4(5), 1992.

[5] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *IN VLDB*, pages 953–964, 2006.

[6] R. Brent. A partnership between biology and engineering. *Nature Biotechnology*, 22:1211–1214, 2004.

[7] C. Bhm, A. Pryakhin, and M. Schubert. The gauss-tree: Efficient object identification in databases of probabilistic feature vectors. In *In Proc. ICDE*, 2006.

[8] A. Carpenter. Software opens the door to quantitative imaging. *Traffic*, 4(2):120–121, 2007.

[9] Center for bioimage informatics. http://bioimage.ucsb.edu/.

[10] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proc. 30th VLDB*, 2004.

[11] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, Oct 2007.

[12] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Trans. on Database Systems*, 21(3):339369, 1996.

[13] S. A. Goff et al. The iplant collaborative: Cyberinfrastructure for plant biology. *Frontiers in Plant Science*, 2(00034), 2011.

[14] Hadoop and HDFS. http://hadoop.org/.

[15] H. G. S. C. International. Finishing the euchromatic sequence of the human genome. *Nature*, 431, October 2004.

[16] iRods: Data grids, digital libraries, persistent archives, and real-time data. http://irods.org/.

[17] K. Kvilekval, D. Fedorov, B. Obara, A. Singh, and B. Manjunath. Bisque: A platform for bioimage analysis and management. *Bioinformatics*, 26(4):544–552, Feb 2010.

[18] L. V. S. Lakshmanan, N. Leone, R. B. Ross, and V. S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Transaction on Database Systems*, 22(3):419469, 1997.

[19] N. Miller, T. D. Brooks, A. Assadi, and E. Spalding. Detection of a gravitropism phenotype in glutamate receptor-like 3.3 mutants of arabidopsis thaliana using machine vision and computation. *Genetics*, 186:585–593, 2010.

[20] NIH data sharing policy. http://grants1.nih.gov/grants/policy/data_sharing/.

[21] Orion project. http://orion.cs.purdue.edu/.

[22] C. Pautasso, O. Zimmermann, and F. Leymann. Restful web services vs. "big"' web services: making the right architectural decision. In *Proc. of 17th Int. Conf. on World Wide Web*, pages 805–814. ACM, 2008.

[23] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *Proc. 33rd Int. Conf. on VLDB*, 2007.

[24] H. Peng. Bioimage informatics: a new area of engineering biology. *Bioinformatics*, 24(17):1827–1836, 2008.

[25] D. Piedrahita et al. Silencing of cdk5 reduces neurofibrillary tangles in transgenic alzheimer's mice. *Journal of Neuroscience*, 30(42):13966–13976, 2010.

[26] C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE'07*, pages 886–895, 2007.

[27] A. D. Sarma, O. Benjelloun, A. Y. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, page 7, 2006.

[28] M. Schilling, A. C. Pfeifer, S. Bohl, and U. Klingmuller. Sharing images. *Standardizing experimental protocols*, 19(4):354–359, 2008.

[29] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, pages 596–605, 2007.

[30] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, jun 1994.

[31] M. Soliman, I. Ilyas, and K. Chang. Top-k query processing in uncertain databases. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 896 –905, april 2007.

[32] J. Swedlow, I. Goldberg, E. Brauner, and P. Sorger. Informatics and quantitative analysis in biological imaging. *Science*, 300(5616):100–102, 2003.

[33] J. Swedlow, S. Lewis, and I. Goldberg. Modelling data across labs, genomes, space and time. *Nature Cell Biology*, 8(11):1190–1194, 2006.

[34] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *Proc. of 31st Int. Conf. on VLDB*, pages 922–933, 2005.

[35] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[36] R. Tuan and C. Lo, editors. *Developmental Biology Protocols, Volume I*, volume 137 of *Methods in Molecular Biology*. Humana Press, November 1999.

[37] M. W. Vannier and R. M. Summers. Sharing images. *Radiology*, 228:23–25, 2003.

[38] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR'05*, pages 262–276, 2005.

Call for Papers

8th International Workshop on Self-Managing Database Systems (SMDB)

In conjunction with
IEEE International Conference on Data Engineering 2013
Brisbane, Queensland, Australia

http://smdb2013.cs.pitt.edu

Important Dates

Paper Submission
November 19, 2012 (abstract)
November 26, 2012 (full paper)

Notification of Acceptance
December 14, 2012

Camera-ready due
December 21, 2012

Workshop
April 8, 2013 (tentative date)

Organization

Workshop Chairs
Alex Labrinidis (U Pittsburgh)
Florian Waas (EMC/Greenplum)

Contact
labrinid@cs.pitt.edu
flw@greenplum.com

Workshop Website
http://smdb2013.cs.pitt.edu

Steering Committee
Data Engineering Work Group on Self-Managing Database Systems

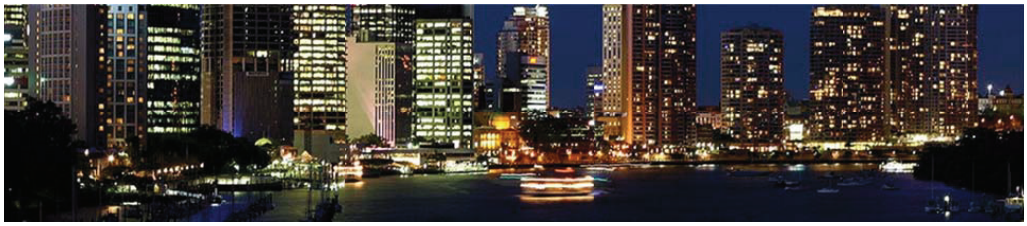
IEEE computer society

The increasing complexity of modern data processing systems calls for self-management capabilities that enable systems to adapt to changing workloads and application scenarios but also facilitate the administration of these systems. In the context of Big Data, the need for such mechanisms is emphasized even more as volume, velocity and variety of data put additional strain on conventional processing platforms.
SMDB is the premier workshop that brings together researchers from academia and industry to exchange ideas, innovations, and experiences around autonomic data management systems and automated administration for data management technology.

Topics of interest include, but are not limited to:

- Principles and architecture of autonomic data management systems
- Self-* capabilities in databases and storage systems
- Data management in cloud and multi-tenant databases
- Autonomic capabilities in database-as-a-service platforms
- Automated physical database design and adaptive query tuning
- Robust query processing techniques
- Self-managing distributed / decentralized / peer-to-peer information systems
- Self-management for Big Data infrastructures
- Monitoring and diagnostics in data management systems
- Policy automation and visualization for datacenter administration
- User acceptance and trust of autonomic capabilities
- Evaluation criteria for self-managing systems
- Use cases and war stories on deploying autonomic capabilities

All aspects of the submission and notification process will be handled electronically. Papers of up to six pages should be submitted via CMT at https://cmt.research.microsoft.com/SMDB2013. Please visit the workshop website for more information and submission guidelines.



Second International Workshop on Data Management in the Cloud (DMC 2013)

In Conjunction with the IEEE International Conference on Data Engineering
April 8-12, 2013 ● Brisbane, Australia

http://db.uwaterloo.ca/dmc2013/

Workshop Co-Chairs
Ashraf Aboulnaga, University of Waterloo
Carlo Curino, Microsoft

Program Committee
Ugur Cetintemel, Brown University
Sameh Elnikety, Microsoft Research
Michael Franklin, UC Berkeley
Hakan Hacigumus, NEC Labs
Mohamed Hefeeda, QCRI
Alfons Kemper, TU Muenchen
Donald Kossmann, ETH Zurich
Tim Kraska, Brown University
Umar Farooq Minhas, Univ of Waterloo
Barzan Mozafari, MIT
Jun Rao, LinkedIn
Kenneth Salem, University of Waterloo
Russell Sears, Microsoft
Khawaja Shams, NASA JPL
Alex Shraer, Google
Adam Silberstein, LinkedIn
Nesime Tatbul, ETH Zurich
Yuan Yu, Microsoft Research

Cloud computing has emerged as a promising computing and business model, delivering unprecedented economic and scalability benefits. At the same time, cloud computing introduces new research challenges, especially for stateful, data-intensive applications. The DMC workshop aims at bringing researchers and practitioners in cloud computing and data management together to discuss research at the intersection of these two fields, and to draw more attention from the larger database and systems research communities to this new and promising field.

Topics of interest include, but are not limited to, the following:

- Elasticity for could data management systems
- Resource and workload management in cloud databases
- Multi-tenancy / isolation
- High availability and reliability in cloud databases
- Cloud computing infrastructure design for cloud data services
- Transactional models for cloud databases
- Virtualization and cloud databases
- Distributed and massively parallel query processing
- Analytics in the cloud
- Storage architectures and technologies for cloud databases
- Privacy and security in cloud data management
- Mobile cloud data management
- Cross-platform interoperability
- Service-level agreements
- Economic/business models and pricing policies
- Novel data-intensive/data-rich computing applications

Important Dates
Paper submission deadline: Thursday, November 1, 2012
Author notification: December 5, 2012
Camera-ready papers due: December 21, 2012

Please visit the workshop web site for detailed instructions on paper submission.

CALL FOR PARTICIPATION

29th IEEE International Conference on Data Engineering
Brisbane, Australia
April 8-12, 2013

www.icde2013.org

The annual ICDE conference addresses research issues in designing, building, managing, and evaluating advanced data-intensive systems and applications. It is a leading forum for researchers, practitioners, developers, and users to explore cutting-edge ideas and to exchange techniques, tools, and experiences.

Venue:
Brisbane is the capital of the Sunshine State, Queensland. It has a population of about 2 million, making it the third-largest city in Australia. With the Gold Coast to the south and the Sunshine Coast to the north, year-round warm climate, spectacular scenery and pleasant locals, Brisbane has lots to offer. The conference will be held at the Sofitel Hotel located within walking distance of Brisbane 's Central Business District.

Affiliated Workshops:
- ICDE Ph.D. Symposium
- Data-Driven Decision Guidance and Support Systems (DGSS)
- Data Engineering Meets the Semantic Web (DESWEB)
- Data Management and Analytics for Semi-Structured Business Processes (DMA4SP)
- Data Management in the Cloud (DMC)
- Graph Data Management: Techniques and Applications (GDM)
- Intelligent Data Processing on Health (IDP)
- Mobile Data Analytics (MoDA)
- Privacy-Preserving Data Publication and Analysis (PrivDB)
- Self-Managing Database Systems (SMDB)

Conference Events:
- Research papers
- Industrial papers
- Demos
- Keynotes
- Tutorials
- Panels
- Workshops

General Chairs:
- Rao Kotagiri *The University of Melbourne, Australia*
- Beng Chin Ooi *National University of Singapore, Singapore*

Program Committee Chairs:
- Christian S. Jensen *Aarhus University, Denmark*
- Chris Jermaine *Rice University, USA*
- Xiaofang Zhou *The University of Queensland, Australia*

IEEE computer society

SAP Microsoft Events Queensland Queensland Government hp

The University Of Queensland AUSTRALIA CSIRO RMIT UNIVERSITY Oracle Labs

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398