Bulletin of the Technical Committee on

# Data Engineering

September 2010 Vol. 33 No. 3

IEEE Computer Society

## Letters

Letter from the Editor-in-Chief . . . . . *David Lomet* 1
Message about the Election of TCDE Chair . . . . . *Erich Neuhold* 2
Information about TCDE Chair Election . . . . . *Carrie Walsh* 2
Candidate Information: David Lomet . . . . . *David Lomet* 3
Report: 5th Int'l Workshop on Self-Managing Database Systems (SMDB 2010) . . *Shivnath Babu, Kai-Uwe Sattler* 4
Letter from the Special Issue Editor . . . . . *Wang-Chiew Tan* 8

## Special Issue on Data Provenance: Applications and New Directions

Provenance in ORCHESTRA . . . . . *Todd J. Green, Grigoris Karvournarakis, Zachary Ives, Val Tannen* 9
Refining Information Extraction Rules using Data Provenance . . . . . *Bin Liu, Laura Chiticariu, Vivian Chu, H.V. Jagadish, Frederick R. Reiss* 17
Socialising Data with Google Fusion Tables . . . . . *Hector Gonzalez, Alon Halevy, Anno Langen, Jayant Madhavan, Rod McChesney, Rebecca Shapley, Warren Shen, Jonathan Goldberg-Kidon* 25
A Rule-Based Citation System for Structured and Evolving Datasets . . . . . *Peter Buneman, Gianmaria Silvello* 33
Panda: A System for Provenance and Data . . . . . *Robert Ikeda, Jennifer Widom* 42
Provenance for Scientific Workflows Towards Reproducible Research . . . . . *Roger Barga, Yogesh Simmhan, Eran Chinthaka, Satya Sahoo, Jared Jackson, Nelson Araujo* 50
Causality in Databases . . . . . *Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y. Halpern, Christoph Koch, Katherine F. Moore, Dan Suciu* 59

## Conference and Journal Notices

SIGMOD 2011 Call for Papers . . . . . 68
PODS 2011 Call for Papers . . . . . 69
ICDE 2011 Conference . . . . . back cover

## Editorial Board

**Editor-in-Chief**

David B. Lomet
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
lomet@microsoft.com

**Associate Editors**

Peter Boncz
CWI
Science Park 123
1098 XG Amsterdam, Netherlands

Brian Frank Cooper
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043

Mohamed F. Mokbel
Department of Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455

Wang-Chiew Tan
IBM Research - Almaden
650 Harry Road
San Jose, CA 95120

---

**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.

## TC Executive Committee

**Chair**

Paul Larson
Microsoft Research
One Microsoft Way
Redmond WA 98052, USA
palarson@microsoft.com

**Vice-Chair**

Calton Pu
Georgia Tech
266 Ferst Drive
Atlanta, GA 30332, USA

**Secretary/Treasurer**

Thomas Risse
L3S Research Center
Appelstrasse 9a
D-30167 Hannover, Germany

**Past Chair**

Erich Neuhold
University of Vienna
Liebiggasse 4
A 1080 Vienna, Austria

**Chair, DEW: Self-Managing Database Sys.**

Anastassia Ailamaki
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland

**Geographic Coordinators**

Karl Aberer (**Europe**)
École Polytechnique Fédérale de Lausanne
Batiment BC, Station 14
CH-1015 Lausanne, Switzerland

Masaru Kitsuregawa (**Asia**)
Institute of Industrial Science
The University of Tokyo
Tokyo 106, Japan

**SIGMOD Liason**

Christian S. Jensen
Department of Computer Science
Åarhus University
DK-8200 Arhus N, Denmark

**Distribution**

Carrie Clark Walsh
IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
CCWalsh@computer.org

# Letter from the Editor-in-Chief

## TCDE Chair Election

There is a letter from the TCDE nominating committee on page two of this issue announcing the election for TCDE Chair and proposing a candidate. In the past, electoral participation has been low. I would urge you, however, to give some thought to whom you would like to see in this position, and make your preference clear by voting in the election. The electoral process is simple and is described in by Carrie Walsh of the IEEE Computer Society in her message. It simply involves having your IEEE CS membership number and visiting a web site to register your vote.

## Workshop Report

Let me draw your attention to the report on the Workshop on Self-Managing Database Systems on page three of this issue. The Self-Managing Database Systems Working Group is a subgroup of the TCDE that focuses on automating the management of database systems. It holds a workshop in conjunction with the International Conference on Data Engineering. The report is on the 2010 workshop, held in Long Beach, California. There will be another of these annual workshops in 2011 in Hanover, Germany in conjunction with ICDE 2011.

## The Current Issue

Data provenance, i.e. information about the source and history of the data, has come to be widely recognized as a vital element for many database applications. While managing data for business data processing purposes has a long and successful history, dealing with provenance is simultaneously relatively new, increasingly important, and both subtle and challenging. Because of this, there is now an active community of researchers who are investigating data provenance, how to uncover it, how to store it with data, how to support, manipulate, annotate and discuss it, and how to propagate it, especially as data from multiple sources are combined and passed onward. Wang-Chiew Tan has worked in this area and knows well the complexities of the area and the researchers in this community. The current issue includes both university research and industrial efforts to deal with provenance. This is the kind of breadth that I like to see in the Bulletin, helping it fulfill its role of bringing current research together with the latest industrial practices to inform our readers in depth on important topics. I want to thank Wang-Chiew for her hard work in successfully bringing this issue on this important topic to fruition. And I thank her for her patience in the somewhat delayed publication process as we awaited details of the TCDE election process.

David Lomet
Microsoft Corporation

## Message about the Election of TCDE Chair

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, Paul Larson, 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 Paul Larson, Erich Neuhold, and Calton Pu is nominating David Lomet for the position as Chair of TCDE. A call for nominations was published in the June issue of the Data Engineering Bulletin but no other nominations were received.

Erich Neuhold
TCDE Nominating Committee Chair

## Information about TCDE Chair Election

The 2010 TCDE Chair election is now open. The Chair's term will run from December 1, 2010 through November 30, 2012. Polls close October 31, 2010, at 5 pm Pacific time. Before entering the poll, please read the following:

- Please take a few moments to review candidate Biosketches and Position Statements in advance of voting here (scroll to bottom of Web page): 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 ).

****Only CS Members can vote. ****
VOTE here: `http://www.surveymonkey.com/s/XDVQB8Z`

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

Carrie Walsh
IEEE Computer Society

# Candidate Information: David Lomet

## Biography

David Lomet is a principal researcher managing Microsoft Research's Database Group. Earlier, he worked at Digital, IBM Research, and Wang Institute. He has a CS Ph.D from U. of Pennsylvania. His research is mostly in database systems, particularly indexing, CCR, and transactions. He has authored over 100 papers (two SIGMOD "best papers") and 40 patents. He is the editor-in-chief of the Data Engineering Bulletin, for which he was given the SIGMOD Contributions Award. He is a fellow of the IEEE, ACM, and AAAS.

## Position statement

The Technical Committee on Data Engineering is the group within the IEEE Computer Society that focuses on data management, including database systems and their use. It sponsors the ICDE conference and co-sponsors a limited number of other conferences and workshops as well. A subgroup of the TC focuses on self-managing databases. The TC's charter is thus an important one, with ICDE being one of the premier database conferences. If elected, I will strive to continue the fine work of my predecessor, Paul Larson, to preserve and enhance these activities. And I will work with Calton Pu, the chair of the ICDE Steering Committee, in his efforts to preserve ICDE as a venue for the publication of great database research.

David Lomet
TCDE Chair Candidate

# Report: 5th Int'l Workshop on Self-Managing Database Systems (SMDB 2010)

Shivnath Babu
Duke University
shivnath@cs.duke.edu

Kai-Uwe Sattler
Ilmenau University of Technology
kus@tu-ilmenau.de

## 1 Introduction

The SMDB Workshop series sponsored by the IEEE TCDE Workgroup on Self-Managing Database Systems brings together researchers and practitioners to exchange ideas related to autonomic data management systems. Previous workshops of the SMDB series focused on core topics in self-managing databases like physical design tuning, problem diagnosis and recovery, and database integration and protection. In addition to core topics, the 2010 workshop aimed to broaden the interest range by covering emerging research areas like Cloud computing, multitenant databases, large-scale storage systems, and datacenter administration.

## 2 Workshop Overview

The 2010 workshop took place on March 1st, 2010 in Long Beach, CA, on the day before ICDE. The workshop received generous sponsorships from Oracle, Sybase iAnywhere, IBM, and Ingres.

The program committee consisted of the members of the SMDB Workgroup's executive committee and other researchers well-known in the area. We received 15 submissions out of which 6 submissions were accepted as long papers and 3 submissions as short papers. In addition to the research paper sessions, the workshop included keynote and panel sessions. The average attendance over all sessions was 35 participants, with the attendance peaking to around 45 during the panel. The full workshop program as well as presentation slides can be found on the workshop's Web site at http://db.uwaterloo.ca/tcde-smdb/smdb10.

## 3 Keynote

Oliver Ratzesberger, Sr. Director Architecture & Operations at eBay Inc., delivered the keynote. Oliver has more than 12 years of experience in large-scale Data Warehousing and Business Intelligence, and is now responsible for the overall eBay site capacity planning and operations cost management – one of the world's largest database infrastructure with more than 20,000 servers.

Oliver's keynote presentation, titled "Agile Enterprise Analytics," gave insights into eBay's new approach of *Analytics as a Service (AaaS)*. AaaS is implemented as virtual data marts which are logical instances created on a shared physical infrastructure. A virtual data mart can be created by a user in less than five minutes by filling a Web form. A virtual data mart may be created for prototyping, development, or testing as well as for production use such as click stream, financial, or performance analytics. The virtual data marts run on a

*Copyright 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

single MPP infrastructure on a single copy of the data, exploiting workload management techniques such as queuing and resource partitioning. The talk concluded with the thesis that agile analytics enables users to try out new ideas much faster and lowers the time to market. Going forward, workload management, scalability, and virtualization will be key factors determining the success of AaaS.

# 4 Paper Sessions

The research paper program was organized into three paper sessions: Indexing and Workload Management, On Cloud and Column Stores, and Query Optimization and Workload Management.

The papers in the first session dealt with index and workload management in database systems. Graefe and Kuno [1] proposed an approach that enables self-selecting and self-tuning indexes for point (equality) queries. Previous work on adaptive indexing had focused on range queries. The new approach uses ideas from database cracking and adaptive merging. Simulation results show that the overhead (the number of records accessed beyond the number of records used in case of full indexes) can be reduced significantly compared to a pure database cracking approach.

Schmidt and Härder [2] presented their work on index selection in native XML databases. Specific challenges in XML indexing include support for different node types, flexible path expressions, and the indexing of both structure and content. After a discussion of storage structures and different XML index types, the authors proposed an approach for estimating the costs and benefits of indexes. Finally, they presented strategies for generating and selecting index candidates. The experimental results showed the benefits of self-tuning indexing as well as the ability to adapt to workload shifts.

Powley et al. [3] dealt with the problem of workload management. They proposed an approach for workload execution control based on query throttling. Here, problematic queries are slowed down to free resources for more important parts of the workload. The authors described three approaches for throttling: a simple controller which uses a diminishing step function, a controller based on a control theoretic approach where the database is treated as a black box, and a hybrid variant that combines these two techniques. The experimental results verified that by using the controllers the important workload can meet desired performance goals continuously.

The main topic of the papers in the second session was data management in the Cloud. Ganapathi et al. [4] introduced a framework for predicting resource requirements for Cloud computing applications. The goal of this work was to predict execution times of MapReduce jobs and Hive queries. For this purpose, the authors used a machine learning technique for correlation analysis between query plan features (such as job configuration parameters like the number of maps and reduces as well as input bytes) and performance metrics (execution time, map time, reduce time, output bytes). Then, these statistical models were used to predict performance metrics for a given setup or to identify workload features which affect performance. Based on these results, the authors discussed the design of a workload generator. At the end of the workshop, this paper was selected by the audience as the most popular paper of this workshop.

Rogers et al. [5] presented a framework that optimizes operational costs of a DBMS on top of an Infrastructure as a Service (IaaS) based on a given query workload and the provider's pricing model while satisfying QoS expectations. They discussed two solutions: a white-box approach with fine-grained estimation of resource consumption and a black-box solution with coarse-grained profiling information. Both approaches were treated as constraint programming problems that were solved using a generic constraint solver. Experimental results using the TPC-H benchmark with PostgreSQL on Amazon EC2 showed the feasibility of the solutions.

Amossen [6] addressed the problem of vertical partitioning in OLTP databases. He presented a model for estimating the cost of executing a workload using a given partitioning schema. Based on the cost model, two algorithms for solving the partition design problem were presented. These algorithms preserve single-sitedness for read queries and allow column replication. The first algorithm uses a quadratic integer programming approach, while the second algorithm is based on simulated annealing. Experiments showed that the latter reduces

the computational effort by 37% for the TPC-C benchmark while returning solutions with costs close to the quadratic approach.

The short paper session contained one paper on query optimization and two papers on workload management. Dash et al. [7] addressed the problem of time-consuming calls to the query optimizer from automated physical design advisors. While the optimizer evaluates each configuration, it creates and evaluates several intermediate plans. Caching partial query plan costs can provide quick answers to subsequent calls to the optimizer. Based on this observation, the PostgreSQL optimizer was extended to support PINUM, an index selection tool for PostgreSQL. Experiments with a synthetic star-schema workload demonstrated 5 times reduction in overhead with only 3% loss in accuracy compared to direct optimizer calls.

The goal of the work presented by Holze et al. [8] was to predict shifts in workloads with a focus on periodic patterns. The proposed approach comprises three steps. The first step involves workload monitoring to collect OS and DBMS metrics as well as SQL statement characteristics. Next, a k-means-based clustering approach is used to derive statement classes. In the third step, the workload is modeled using n-gram models as an approximation of Markov models. Detecting that the current workload is similar to a workload $W$ observed in the past allows the system to switch to a a pre-optimized configuration for $W$.

Finally, Abouzour et al. [9] described a multi-input single-output controller to determine the optimal multiprogramming level (MPL) in Sybase. The throughput level (number of server-side requests), the total number of outstanding requests, and the control interval are used as input. The output parameter is the MPL to use in the next control interval. The tuning task was implemented using different algorithms: hill climbing, global parabola approximation, as well as a hybrid approach that switches between the other two strategies. Experimental results showed that the hybrid approach was able to reach around 90% of the optimal (hand-tuned) value.

# 5 Panel

The workshop concluded with a panel on "Databases, MapReduce, and the Cloud–Oh My! What's in it for the Administrator?", with six distinguished panelists: Ashraf Aboulnaga from Univ. of Waterloo, Namit Jain from Facebook, Guy Lohman from IBM, Oliver Ratzesberger from eBay, Benjamin Reed from Yahoo!, and Jingren Zhou from Microsoft.

The computer science community has witnessed a recent debate on the performance advantages and disadvantages of database systems vs. MapReduce systems for large-scale analytics. The panelists were asked to discuss the system administration aspects of these two systems. Questions posed to the panelists included: (a) are there fundamental strengths or weaknesses that databases and MapReduce systems have regarding administration? (b) what does administering a MapReduce system involve? (c) does the emergence of Cloud computing shift the "cost of administration" equation in favor of either of these systems? (d) what lessons should MapReduce systems learn regarding administration from database systems? (e) will the roles of the data analyst, system developer, and system administrator increasingly overlap in the future?

Aboulnaga discussed how the number of software and hardware instances that have to be administered in a Cloud setting is very high, and the impact of an administrator mistake can be high as well. Administrative tasks like upgrades are hard on the Cloud. Aboulnaga also discussed how database and MapReduce systems can learn good practices from each other. For example, databases need to learn how to manage data that is not already loaded into the system, while MapReduce systems can learn how to auto-tune configuration parameters.

Jain gave an overview of Facebook's 12 PetaByte data analytics infrastructure driven by Hadoop. This warehouse runs on 9600 cores, and absorbs 10 TeraBytes (compressed) of new data per day. Data analysts and engineers use a SQL-like language to run declarative queries in this massive warehouse. Jain mentioned the challenges in this setting which include ensuring performance isolation across jobs in a shared cluster, metadata discovery, cluster and job monitoring, and efficient query plan generation.

Lohman summarized the Cloud's main value propositions from a customer's perspective which include pay-as-you-go usage and reduction of human effort. A study done by IBM found that Cloud deployment solutions

can liberate considerable amounts of IT funds for new development. However, Lohman questioned whether these benefits will continue to be realized years from now when the Cloud becomes legacy. He warned of a future where the benefits of Cloud computing disappear due to improper management of mixed workloads, schema, service-level objectives, and security.

Ratzesberger developed on his keynote theme emphasizing the need for Analytics as a Service, and the accompanying management challenges. Virtual data marts have to be provisioned automatically on demand based on user specification, while addressing workload management challenges like performance isolation in a shared physical cluster.

Reed identified three current administrative roles in Hadoop at Yahoo!: developers who write Hadoop code with certain tuning knobs, operations personnel who configure Hadoop clusters but do not configure job-level tuning knobs, and release engineering personnel who prepare Hadoop images for release. Since interactions among people in these roles is limited, Reed opined that Hadoop stands to benefit immensely from auto-tuning techniques.

Zhou observed that while database systems run on many tens of machines, MapReduce systems run on tens of thousands of machines. While database administrators focus on issues like physical design tuning and system recovery, in large-scale systems running on the Cloud, most of an administrator's time today goes into keeping the system and data alive and available for use.

## 6 SMDB 2011

SMDB 2011 is being organized by Vivek Narasayya and Neoklis Polyzotis, and is scheduled for April 10th, 2011 in Hannover, Germany. Visit `http://db.uwaterloo.ca/tcde-smdb/smdb11` for more information.

## References


[1] Goetz Graefe, Harumi Kuno: Adaptive Indexing for Relational Keys. In *Proc. ICDE Workshops 2010*, pp. 69-74, 2010.

[2] Karsten Schmidt, Theo Härder: On the Use of Query-Driven XML Auto-Indexing. In *Proc. ICDE Workshops 2010*, pp. 81-86, 2010.

[3] Wendy Powley, Patrick Martin, Mingyi Zhang, Paul Bird, Keith McDonald: Autonomic Workload Execution Control Using Throttling. In *Proc. ICDE Workshops 2010*, pp. 75-89, 2010.

[4] Archana Ganapathi, Yanpei Chen, Armando Fox, Randy Katz, David Patterson: Statistics-Driven Workload Modeling for the Cloud. In *Proc. ICDE Workshops 2010*, pp. 87-92, 2010.

[5] Jennie Rogers, Olga Papaemmanouil, Ugur Cetintemel: A Generic Auto-Provisioning Framework for Cloud Databases. In *Proc. ICDE Workshops 2010*, pp. 63-68, 2010.

[6] Rasmus Resen Amossen: Vertical Partitioning of Relational OLTP Databases Using Integer Programming. In *Proc. ICDE Workshops 2010*, pp. 93-98, 2010.

[7] Debabrata Dash, Ioannis Alagiannis, Cristina Maier, Anastasia Ailamaki: Caching All Plans with Just One Optimizer Call. In *Proc. ICDE Workshops 2010*, pp. 105-110, 2010.

[8] Marc Holze, Ali Haschimi, Norbert Ritter: Towards Workload-Aware Self-Management: Predicting Significant Workload Shifts. In *Proc. ICDE Workshops 2010*, pp. 111-116, 2010.

[9] Mohammed Abouzour, Kenneth Salem, Peter Bumbulis: Automatic Tuning of the Multiprogramming Level in Sybase SQL Anywhere. In *Proc. ICDE Workshops 2010*, pp. 99-104, 2010.

## Letter from the Special Issue Editor

The importance of data provenance has been increasingly recognized by both users and publishers of data. For users of data, the scientific basis of their analysis relies largely on the credibility and trusthworthiness of their input data. For publishers of data, the provision of provenance as part of their published data is important for scholarship and reproducibility. In today's Internet era, where complex ecosystems of data are even more prevalent, it is no wonder that data provenance has become a major research topic in many conferences and workshops. Data provenance was already the topic of the December 2007 issue of IEEE Data Engineering Bulletin. This issue attempts to complement the December 2007 issue by focusing on how provenance has been captured and exploited by systems that either have been developed or are in the process of being developed in the industry and the academia, and by reporting on new interesting research directions.

The first three articles describe how provenance has been applied in systems developed in the industry and academia. *Provenance in ORCHESTRA*, by Green *et al.*, describes a collaborative data sharing system ORCHESTRA, whose architecture and provenance model are largely motivated from the needs of the life sciences community. As part of ORCHESTRA, the authors describe an interesting application of provenance to the specification of trust policies and update exchange. In the article titled *Refining Information Extraction Rules using Data Provenance*, Liu *et al.* describe how provenance can be used to aid the process of developing information extraction rules. Here, provenance is exploited to understand the changes that need to be made to the information extraction rules so that the result is what the user desires. Gonzalez *et al.* describe how Google Fusion Tables can be used to faciliate collaboration around data sets in their article *Socializing Data with Google Fusion Tables*. Attribution and provenance are anticipated to play a significant role in socializing data and this article provides an accessible exposition to the attribution feature of Google Fusion Tables, as well as the authors' experience with the system.

It is important for scientists to be able to pinpoint and cite data easily for proper scholarship. In the article *A Rule-Based Citation System for Structured and Evolving Datasets*, Buneman and Silvello discuss the requirements that must be met by a citation system in order to guarantee consistency and integrity of citations. The issues that arise in two data sets found in the fields of Digital Libraries and Curated Databases, as well as a rule-base system for automatically generating proper citations that addresses the issues are fully described in this article.

According to the December 2007 issue of IEEE Data Engineering Bulletin, connecting workflow and data provenance is said to be "the most interesting research challenge in the general field of provenance models". In *Panda: A System for Provenance and Data*, Ikeda and Widom report on their work-in-progress Panda project, whose goal is to develop a system that would seamlessly manage and query both data-based and process-based provenance. The authors have provided a simple and yet elaborate example for explaining various aspects of their vision, as well as the challenges associated with building such a system.

As scientists tend to follow the "extract, transform, and analyze" workflow model to process scientific data, many workflow systems aim to provide scientists with a platform for easily authoring workflows. The Trident scientific workflow system is described in the article *Provenance for Scientific Workflows Towards Reproducible Research*. Barga *et al.* give a good overview of various provenance-related components of Trident. An integral part of Trident is the automatic capturing of provenance in both control-flow and data-flow aspects of workflows, as well as in workflow evolution.

Finally, the article *Causality in Databases* takes a fresh perspective at the problem of explaining outputs of database transformations. Meliou *et al.* give an accessible description of how the notions of causality and responsibility, which are well-studied notions in philosophy, can be applied to databases to provide explanations of answers to database transformations, and point out several promising research directions.

Wang-Chiew Tan
IBM Almaden Research Center and UC Santa Cruz

# Provenance in ORCHESTRA

Todd J. Green*
University of California, Davis
Davis, CA USA
green@cs.ucdavis.edu

Grigoris Karvounarakis*
LogicBlox ICS-FORTH
Atlanta, GA Heraklion, Greece
gregkar@gmail.com

Zachary G. Ives Val Tannen
University of Pennsylvania
Philadelphia, PA USA
{zives,val}@cis.upenn.edu

**Abstract**

*Sharing structured data today requires agreeing on a standard schema, then mapping and cleaning all of the data to achieve a single queriable mediated instance. However, for settings in which structured data is collaboratively authored by a large community, such as in the sciences, there is seldom consensus about how the data should be represented, what is correct, and which sources are authoritative. Moreover, such data is dynamic: it is frequently updated, cleaned, and annotated. The* ORCHESTRA *collaborative data sharing system develops a new architecture and consistency model for such settings, based on the needs of data sharing in the life sciences. A key aspect of* ORCHESTRA*'s design is that the provenance of data is recorded at every step. In this paper we describe* ORCHESTRA*'s provenance model and architecture, emphasizing its integral use of provenance in enforcing trust policies and translating updates efficiently.*

## 1 Introduction

One of the most elusive goals of the data integration field has been supporting sharing across large, heterogeneous populations. While data integration and its variants (e.g., data exchange [9] and warehousing) are being adopted in the enterprise, little progress has been made in integrating broader communities. Yet the need for sharing data across large communities is increasing: most of the physical and life sciences have become data-driven as they have attempted to tackle larger questions. The field of bioinformatics, for instance, has a plethora of different databases, each providing a different perspective on a collection of organisms, genes, proteins, diseases, and so on. Associations exist between the different databases' data (e.g., links between genes and proteins, or gene homologs between species). Unfortunately, data in this domain is surprisingly difficult to integrate, primarily because conventional data integration techniques require the development of a single global schema and complete global data consistency. Designing one schema for an entire community like systems biology is arduous, involves many revisions, and requires a central administrator.

Even more problematic is the fact that the data or associations in different databases are often contradictory, forcing individual biologists to choose values from databases they personally consider most authoritative or trusted [23]. Such inconsistencies are not handled by data integration tools, since there is no consensus or "clean" version of the data. Thus, scientists simply make their databases publicly downloadable, so users can

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

*Work performed while at the University of Pennsylvania

copy and convert them into a local format (using ad hoc scripts). Meanwhile the original data sources continue to be edited. In some cases the data providers publish weekly or monthly lists of updates (*deltas*) to help others keep synchronized. Today, few participants, except those with direct replicas, can actually exploit such deltas; hence, once data has been copied to a dissimilar database, it begins to diverge from the original.

In order to provide collaborating scientists, organizations, and end users with the tools they need to share and revise structured data, our group has developed a new architecture we term *collaborative data sharing systems* [18] (CDSSs) and the first implementation of a CDSS in the form of the ORCHESTRA system. The CDSS provides a principled semantics for exchanging data and updates among autonomous sites, which extends the data integration approach to encompass scientific data sharing practices and requirements—in a way that also generalizes to many other settings. The CDSS models the exchange of data among sites as *update exchange* among peers, which is subject to transformation (via schema mappings), filtering (based on trust policies regarding source authority), and local revision or replacement of data. As a prerequisite to assessing trust, the CDSS records the derivation of all exchanged data, i.e., its *provenance* or lineage [5].

In this paper, we provide an overview of the basic operation of the ORCHESTRA CDSS, emphasizing its use of data provenance for enforcing trust policies and for performing update exchange incrementally and efficiently. This paper summarizes our main results from [14, 20, 21]; we mention further aspects of the system in Section 6.

## 2 Overview of CDSS and ORCHESTRA

The CDSS model represents a natural next step in the evolution of ideas from data integration, peer data management systems [17] (PDMS), and data exchange [9]. The PDMS model removes data integration's requirement of a single central schema: rather, the PDMS supports greater flexibility and schema evolution through *multiple* mediated schemas (peers) interrelated by compositional schema mappings. Along a different dimension, data exchange alleviates the tight coupling between data sources and queriable target data instance: it enables *materialization* of data at the target, such that queries over the target will be given the same answers as in traditional virtual data integration.

As in the PDMS, the CDSS allows each data sharing participant to have an independent schema, related to other schemas via compositional schema mappings. However, in a CDSS each participant (peer) materializes its own database instance (as in data exchange). Through a variety of novel techniques we sketch in this paper and describe in detail in [14, 20, 21], the CDSS enables the peer to *autonomously control* what data is in the instance by applying updates and policies about which data is most trusted.[1]

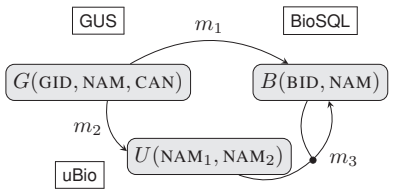
### 2.1 The topology of data sharing: specifying how peers are related

A CDSS collaboration begins with a set of *peer* databases, each with its own schema and local data instance. Each peer's users pose queries and make updates directly to its local data instance.

**Example 1:** Consider (see Figure 1) a bioinformatics collaboration scenario based on databases of interest to affiliates of the Penn Center for Bioinformatics. GUS, the Genomics Unified Schema covers gene expression, protein, and taxon (organism) information; BioSQL, affiliated with the BioPerl project, covers very similar concepts; and a third schema, uBio, establishes synonyms among taxa.[2] Instances of these databases contain taxon information that is autonomously maintained but of mutual interest to the others. For the purposes of our example we show only one relational table in each of the schemas, as follows. Peer GUS associates taxon identifiers, scientific names, and what it considers *canonical* scientific names via relation $G(\text{GID}, \text{NAM}, \text{CAN})$; peer BioSQL associates its own taxon identifiers with scientific names via relation $B(\text{BID}, \text{NAM})$; and peer uBio records synonyms of scientific names via relation $U(\text{NAM}_1, \text{NAM}_2)$. □

[1] Our implementation assumes relational schemas, but the CDSS model extends naturally to other data models such as XML [10].

[2] The real databases can be found at `http://www.gusdb.org`, `http://bioperl.org`, and `http://www.ubio.org`.




$$(m_1) \quad G(g,n,c) \rightarrow \exists b\, B(b,n)$$
$$(m_2) \quad G(g,n,c) \rightarrow U(n,c)$$
$$(m_3) \quad B(b,n_1) \wedge U(n_1,n_2) \rightarrow B(b,n_2)$$

Figure 1: Mappings among three bioinformatics databases

$G$ :

| GID | NAM | CAN |
|---|---|---|
| 828917 | *Oscinella frit* | *Drosophila melanogaster* |
| 2616529 | *Musca domestica* | *Musca domestica* |

$U$ :

| $\text{NAM}_1$ | $\text{NAM}_2$ |
|---|---|

$B$ :

| BID | NAM |
|---|---|
| 4472 | *Periplaneta americana* |

(a) Tables from GUS, uBio, and BioSQL before update exchange. ($U$ is empty.)

$U$ :

| $\text{NAM}_1$ | $\text{NAM}_2$ |
|---|---|
| *Oscinella frit* | *Drosophila melanogaster* |
| *Musca domestica* | *Musca domestica* |

$B$ :

| BID | NAM |
|---|---|
| 4472 | *Periplaneta americana* |
| $\perp_1$ | *Oscinella frit* |
| $\perp_1$ | *Drosophila melanogaster* |
| $\perp_2$ | *Musca domestica* |

(b) Tables after update exchange. Shaded rows indicate newly-inserted tuples. ($G$ is unchanged.)

Figure 2: Bioinformatics database instances before and after update exchange

The participants of the CDSS collaboration specify relationships among their databases using *schema mappings*.

**Example 2:** Continuing with Example 1, suppose it is agreed in this collaboration that certain data in GUS should also be in BioSQL. This represented in Figure 1 by the arc labeled $m_1$. The specification $G(g,n,c) \rightarrow \exists b\, B(b,n)$ associated with $m_1$ is read as follows: if $(g,n,c)$ is in table $G$, the value $n$ must also be in some tuple $(b,n)$ of table $B$, although the value $b$ in such a tuple is not determined. The specification just says that there must be such a $b$ and this is represented by the existential quantification $\exists b$. Here $m_1$ is an example of *schema mapping*. Two other mappings appear in Figure 1. Peer uBio should also have some of GUS's data, as specified by $m_2$. Mapping $m_3$ is quite interesting: it stipulates data in BioSQL based on data in uBio but also on data *already* in BioSQL. As seen in $m_3$, relations from multiple peers may occur on either side. We also see that individual mappings can be "recursive" and that cycles are allowed in the graph of mappings. □

Schema mappings (expressed in ORCHESTRA using the well-known formalism of *tuple-generating dependencies* or tgds [2]) are logical assertions that the data instances at various peers are expected to jointly *satisfy*. As in data exchange [9], a large class of mapping graph cycles can be handled safely, while certain complex examples cause problems and are prohibited.

In summary, every CDSS specification begins with a collection of peers/participants, each with its own relational schema, and a collection of schema mappings among peers. Like the schemas, the mappings are designed by the administrators of the peers to which data is to be imported. (In addition, administrators also supply *trust policies*, described in Section 3.) By joining ORCHESTRA, the participants agree to share the data from their local databases. Mappings and trust policies provide a more detailed declarative specification of how data is to be shared.

## 2.2 The semantics of query answers, and what must be materialized at each peer

Given a CDSS specification of peers and schema mappings, the question arises of how data should be propagated using the declarative mappings, and what should be materialized at the target peer. Clearly, a user query posed over a peer's materialized instance should provide answers using relevant data from *all* the peers. This requires

materializing at every peer an instance containing not only the peer's locally contributed data, but also additional facts that *must* be true, given the data at the other peers along with the constraints specified by the mappings. However, while the mappings relating the peers tell us which peer instances are together considered "acceptable," they do not fully specify the complete peer instances to materialize.

In CDSS we follow established practice in data integration, data exchange and incomplete databases [1, 9] and use *certain answers* semantics: a tuple is "certain" if it appears in the query answer no matter what data instances (satisfying the mappings) we apply the query to. In virtual data integration, the certain answers to a query are computed by reformulating the query across all peer instances using the mappings. In CDSS, as in data exchange, we materialize special local instances that can be used to compute the certain answers. This makes query evaluation a fast, local process. We illustrate this with our running example.

**Example 3:** Suppose the contents of $G$, $U$, and $B$ are as shown in Figure 2(a). Note that the mappings of Figure 1 are not satisfied: for example, $G$ contains a tuple $(828917, \text{"}\textit{Oscinella frit}\text{"}, \text{"}\textit{Drosophila melanogaster}\text{"})$ but $B$ does not contain any tuple with "*Oscinella frit*" which is a violation of $m_1$. Let us patch this by adding to $B$ a tuple $(\perp_1, \text{"}\textit{Oscinella frit}\text{"})$ where $\perp_1$ represents the unknown value specified by $\exists b$ in the mapping $m_1$. We call $\perp_1$ a *labeled null*. Adding just enough patches to eliminate all violations results in the data in Figure 2(b).[3] □

In order to support edits and accommodate disagreement among participants, CDSS incorporates mechanisms for *local curation* of database instances. This allows CDSS users to modify or delete *any* data in their local database, even data that has been imported from elsewhere via schema mappings. In this way, CDSS users retain full control over the contents of their local database. The technical obstacle in supporting such a feature is how to preserve a notion of semantic consistency with respect to the mappings, when such modifications may introduce violations of the mappings.

**Example 4:** Refer again to Figure 2(b), and suppose that the curator of BioSQL decides that she is not interested in house flies, and wishes to delete tuple $B(\perp_2, \text{"}\textit{Musca domestica}\text{"})$ from her local instance. Since, as we have seen, this tuple was added to satisfy a mapping, the deletion of the tuple leads to a violation of some mapping. □

To allow such local curation, CDSS stores deleted tuples in *rejection tables*, and newly inserted tuples in *local contribution tables*, and converts user-specified mappings into internal mappings that take the contribution and rejection tables explicitly into account. Alternatively, sometimes local curation corrects *mistakes* in the imported data; if we wish for the corrections to be propagated back to the original source tuples, ORCHESTRA also incorporates facilities for *bidirectional mappings* [20].

# 3 Trust policies and provenance

CDSS participants often need to filter data based how much they trust it (and its sources). In order to allow the specification of such *provenance-based trust policies*, ORCHESTRA records the *provenance* of exchanged data, i.e., "how data was propagated" during the exchange process.

The most intuitive way to picture CDSS provenance is as a graph having two kinds of nodes: *tuple nodes*, one for each tuple in the system, and *mapping nodes*, where several such nodes can be labeled by the same mapping name. Edges in the graph represent *derivations* of tuples from other tuples using mappings. Special mapping nodes labeled "+" are used to identify original source tuples.

**Example 5:** Refer to Figure 3, which depicts the provenance graph corresponding to the bioinformatics example from Figure 2 (we have abbreviated the data values to save space). Observe there is a "+" mapping node pointing to $G(26, \textit{Musc.}, \textit{Musc.})$; this indicates that it is one of the source tuples from Figure 2(a), present before

[3] Note that sometimes patching one violation may introduce a new violation, which in turn must be patched; hence this process is generally an iterative one, however under certain constraints it always terminates.

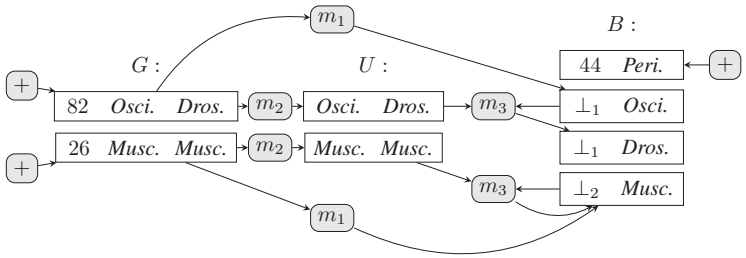



Figure 3: Provenance graph for bioinformatics example

```
EVALUATE TRUST OF {
    FOR [B $x] <$p [] <-+ [$y]
    WHERE $p = m1 OR $p = m3
    INCLUDE PATH [$x] <-+ [$y]
    RETURN $x
} ASSIGNING EACH leaf_node $y {
    CASE $y in G and
        $y.n = "Musc." : SET false
    DEFAULT : SET true
} ASSIGNING EACH mapping $p($z) {
    CASE $p = m3 : SET false
    DEFAULT : set $z
}
```

Figure 4: A provenance query in ProQL

update exchange was performed. Next, observe that $G(26, \textit{Musc.}, \textit{Musc.})$ is connected to $U(\textit{Musc.}, \textit{Musc.})$ via a mapping node labeled $m_2$. This indicates that $U(\textit{Musc.}, \textit{Musc.})$ was derived using $G(26, \textit{Musc.}, \textit{Musc.})$ with $m_2$. Also, notice that $B(\perp_2, \textit{Musc.})$ has a derivation from $G(26, \textit{Musc.}, \textit{Musc.})$ via mapping $m_1$. Finally, note that $B(\perp_2, \textit{Musc.})$ also has a second derivation, from itself and $U(\textit{Musc.}, \textit{Musc.})$ via mapping $m_3$. The graph is thus cyclic, with tuples involved in their own derivations. In general, when mappings are recursive, the provenance graph may have cycles. □

Provenance graphs form the basis of our ORCHESTRA implementation, and [14] shows how they can be computed during data exchange and stored together with exchanged data. Underlying the graphical model, however, is another, equivalent perspective on CDSS provenance, based on a framework of *semiring-annotated relations* [15, 10]. These are algebraic structures that arise naturally with database provenance. Their two operations correspond to joint use of data (represented in the graph model by the incoming edges into a mapping node) and to alternative use of data (represented in the graph model by the incoming edges into a tuple node). $\mathcal{M}$*-semirings* [19] extend those structures with unary functions to capture mappings. The semiring framework has the virtue of uniformly capturing as special cases many other data provenance models from the literature (e.g., *lineage* [7], *why-provenance* [4], and *Trio lineage* [3]). By putting all these models on equal footing, we are able to make precise comparisons of various kinds: e.g., their relative *informativeness*, or their interactions with *query optimization* [12]. In particular, we can explain precisely why ORCHESTRA's provenance is strictly more informative than the other models. Moreover, we see that ORCHESTRA provenance captures the most general semiring computations and thus can be specialized to compute events (hence probabilities), scores, counts, security access levels, etc, as we discuss in Section 5 and [21]. Provenance annotations also assist in formulating efficient algorithms for update exchange, as we explain in Section 4 and [14, 20].

Using such data provenance information, we can compute trust scores for exchanged data, based on CDSS users' beliefs about the trustworthiness of source data and mappings. Trust policies can then be expressed to control the flow of data through mappings, allowing CDSS administrators to specify which tuples are "trusted" or "distrusted," depending on their provenance.

**Example 6:** Some possible trust policies in our bioinformatics example:

- Peer BioSQL distrusts any tuple $B(b, n)$ if the data came from GUS and $n =$ "*Musca domestica,*" and trusts any tuple from uBio.
- Peer BioSQL distrusts any tuple $B(b, n)$ that came from mapping $m_3$. □

Once specified, the trust policies are incorporated into the update exchange process: when the updates are being translated into a peer P's schema they are accepted or rejected based on P's trust conditions.

The example above illustrates *Boolean trust policies*, which classify all tuples as either (completely) trusted or (completely) untrusted, depending on their provenance and contents. In fact, CDSS also allows richer forms of *ranked trust policies* in which *trust scores* are computed for tuples indicating various "degrees of trust." When a conflict is detected among data from multiple sources (for example, by a primary key violation), these scores

can be used to resolve the conflict by selecting the tuple with the highest trust score and discarding those with which it conflicts. Fortunately, our same semiring-based provenance information can be used to evaluate either Boolean or ranked trust, as well as—possibly contradictory—trust policies of different CDSS peers without recomputing data exchange solutions [21].

# 4 Dynamic operation and update exchange

Given that ORCHESTRA performs query answering on locally materialized data instances, this raises the question of data freshness in the peer instances. The CDSS approach sees data sharing as a fundamentally *dynamic* process, with frequent "refreshing" *data updates* that need to be propagated efficiently. This process of dynamic update propagation is called *update exchange*. It is closely related to the classical *view maintenance problem* [16].

Operationally, CDSS functions in a manner reminiscent of *revision control systems*, but with peer-centric conflict resolution strategies. The users located at a peer P query and update the local instance in an "offline" fashion. Their updates are recorded in a *local edit log*. Periodically, upon the initiative of P's administrator, P requests that the CDSS perform an update exchange operation. This *publishes* P's local edit log, making it globally available via central or distributed storage [27]. This also subjects P to the effects of the updates that the other peers have published (since the last time P participated in an update exchange). To determine these effects, the CDSS performs incremental *update translation* using the schema mappings to compute corresponding updates over P's local instance: the translation finds matches of incoming tuples to the mappings' sources, and applies these matchings to the mappings' targets to produce outgoing tuples (recall the "patching" process in Example 3).

Doing this on updates means performing data exchange incrementally, with the goal of maintaining peer instances satisfying the mappings. Therefore, in CDSS the mappings are more than just static specifications: they enable the dynamic process of *propagating* updates.

**Example 7:** Refer again to Figure 2(b), and suppose now that the curator of uBio updates her database by adding another synonym for the fruit fly: $U($"*Oscinella frit*", "*Oscinella frit Linnaeus*"$)$. When this update is published, it introduces a violation of mapping $m_3$ that must again be "patched." The update translation process therefore inserts a corresponding tuple into BioSQL: $B(\perp_1,$ "*Oscinella frit Linnaeus*"$)$. □

Propagating the effects of insertions to other peers is a straightforward matter of applying techniques from [16]. Deletions are more complex, both in terms of propagating them *downstream* (to instances mapped from the initial deletion) and optionally *upstream* (to sources of a derived, now-deleted tuple).

*Downstream propagation* involves identifying which tuples in a derived instance are dependent on the tuple(s) we are deleting and have no alternate derivations from the remaining source tuples. The algorithms of [16] provide one solution, involving deleting and attempting to re-derive all dependent tuples. Provenance enables a more efficient algorithm [14], which determines if a tuple has alternative derivations, avoiding unnecessary deletions and rederivations.

*Upstream propagation* requires determining the sources of a derived tuple, and removing (some of) them, so that this tuple is no longer derivable through the mappings. This problem is closely related to the *view update problem* [8], and raises the possibility of *side effects*: if a source tuple is removed, additional tuples that were derived from it and were not among the users' deletions may also be deleted. Traditionally, we disallow updates whenever the database integrity constraints do not guarantee side-effect-free behavior (a very restrictive approach in practice). In ORCHESTRA we provide a more flexible approach. We use provenance to check at *run-time* whether some deletion propagation would cause side effects to any other peer specified by the user. We only propagate to the sources those deletions that do not cause side effects, while we handle the remaining deletions through the rejection tables (Section 2.2). This allows much greater levels of update propagation in practice.

# 5 Querying provenance

To this point, we have described the *internal* uses of provenance in a CDSS. However, provenance is also useful to CDSS end-users as they perform data curation and other manual tasks. Unfortunately, the complexity of provenance graphs can often make it difficult for such users to explore the provenance of data items they are interested in. Moreover, curators may not be interested in the complete provenance of some items, but only derivations involving certain mappings or peers they consider authoritative.

For these reasons, we have designed and implemented *ProQL* [21], a query language for provenance. ProQL can help curators or other CDSS users explore and navigate through provenance graphs, based on as little information as they have in their disposal, through the use of *path expressions*. It also allows them to focus only on parts of the graph which are of interest to them. ProQL builds upon the fact that our provenance model generalizes a variety of annotation computations, such as trust scores, probabilities, counts, security access levels or derivability. In particular, ProQL allows CDSS users to specify "source" annotations for tuple and mapping nodes in these graphs and uses those to compute annotations for derived tuples of interest.

**Example 8:** The query shown in Figure 4 illustrates some of these features. First, it matches all derivations of tuples in $B$ (of any length, as indicated by $\leftarrow$+) whose last step involves mappings $m_1$ and $m_3$. Then, it specifies that any derivations through $m_1$ as well as tuples in $G$ with name "*Musc.*" are untrusted, while everything else is trusted (essentially, these are the trust policies from Example 6). Applied on the provenance graph of Figure 3, it determines that the tuples $B(44, \text{"}\textit{Peri.}\text{"})$, $B(\bot_1, \text{"}\textit{Osci.}\text{"})$ are trusted, while the remaining tuples in $B$ are untrusted. □

More details about the syntax of ProQL, as well as indexing techniques to optimize provenance query processing can be found in [21].

# 6 Related work

In this paper we surveyed briefly the core update exchange and provenance facilities of ORCHESTRA, as originally presented in [14, 20], and the provenance query language of [21]. Other major features of the system include a distributed conflict reconciliation algorithm [27], a distributed storage and query engine [28], a keyword-based query system incorporating rankings and user feedback [25]. Related work on incremental update optimization appears in [13]. Data sharing with conflict update resolution policies is studied in [22]. The semiring framework for ORCHESTRA's provenance model was introduced in [15], and further elaborated in [10] and [12].

With hindsight, we find misleading the term "how-provenance" (originating in a footnote of [15] and popularized in [5]). It allows for an interpretation according to which this is yet another provenance model, parallel to why- and where-provenance [4]. As we saw, the semiring framework encompasses many previously proposed provenance models. This includes where-provenance as soon as we annotate other data elements beyond tuples, as shown in [10] (see also [26]).

Recognizing the limitations of why-provenance, [6] introduces *route-provenance*, which is closest to the provenance model used in ORCHESTRA, albeit used for a different purpose—debugging schema mappings. Our model maintains a graph from which provenance can be incrementally recomputed or explored, whereas in [6] the shortest route-provenances are recomputed on demand. [5] contains many more references on provenance that space does not allow us to mention here.

ORCHESTRA builds upon foundations established by PDMS (e.g., [17]) and data exchange [24, 9]. In [11], the authors use target-to-source tgds to express trust. We support a more flexible trust model where each peer may express different levels of trust for other peers, and trust conditions compose along paths of mappings. Moreover, our approach does not increase the complexity of computing a solution.

# 7 Conclusions

The ORCHESTRA project represents a re-thinking of how data should be shared at large scale, when differences of opinion arise not only in the data representation, but also which data is correct. It defines new models and algorithms for update exchange, provenance, trust, and more. Our initial prototype system demonstrates the feasibility of the CDSS concept, and we are releasing it into open source at `code.google.com/p/penn-orchestra`. We believe that many opportunities for further research are enabled by our platform. For example, we believe there are many interesting avenues of exploration along derivations, conflicting data, data versions, etc. We also feel it would be worthwhile to explore integrating probabilistic data models into the CDSS architecture.

**Acknowledgements.** We thank the other members of the ORCHESTRA team, particularly Nicholas Taylor, Olivier Biton, and Sam Donnelly, for their contributions to the effort; and the Penn Database Group and Wang-Chiew Tan for their feedback. This work was funded in part by NSF IIS-0447972, IIS-0513778, IIS-0713267, and CNS-0721541, and DARPA HRO1107-1-0029.

# References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *PODS*, Seattle, WA, 1998.

[2] C. Beeri and M. Vardi. A proof procedure for data dependencies. *JACM*, 31(4), October 1984.

[3] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, 2006.

[4] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *ICDT*, 2001.

[5] J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4), 2009.

[6] L. Chiticariu and W.-C. Tan. Debugging schema mappings with routes. In *VLDB*, 2006.

[7] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM TODS*, 25(2), 2000.

[8] U. Dayal and P. A. Bernstein. On the correct translation of update operations on relational views. *TODS*, 7(3), 1982.

[9] R. Fagin, P. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *TCS*, 336, 2005.

[10] J. N. Foster, T. J. Green, and V. Tannen. Annotated XML: Queries and provenance. In *PODS*, 2008.

[11] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W.-C. Tan. Peer data exchange. In *PODS*, 2005.

[12] T. J. Green. Containment of conjunctive queries on annotated relations. In *ICDT*, 2009.

[13] T. J. Green, Z. G. Ives, and V. Tannen. Reconcilable differences. In *ICDT*, 2009.

[14] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB*, 2007.

[15] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, Beijing, China, June 2007.

[16] A. Gupta and I. S. Mumick, editors. *Materialized Views: Techniques, Implementations and Applications*. The MIT Press, 1999.

[17] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, March 2003.

[18] Z. Ives, N. Khandelwal, A. Kapur, and M. Cakir. ORCHESTRA: Rapid, collaborative sharing of dynamic data. In *CIDR*, January 2005.

[19] G. Karvounarakis. *Provenance in Collaborative Data Sharing*. PhD thesis, University of Pennsylvania, 2009.

[20] G. Karvounarakis and Z. G. Ives. Bidirectional mappings for data and update exchange. In *WebDB*, 2008.

[21] G. Karvounarakis, Z. G. Ives, and V. Tannen. Querying data provenance. In *SIGMOD*, 2010.

[22] L. Kot and C. Koch. Cooperative update exchange in the Youtopia system. In *Proc. VLDB*, 2009.

[23] P. Mork, R. Shaker, A. Halevy, and P. Tarczy-Hornoch. PQL: A declarative query language over dynamic biological schemata. In *American Medical Informatics Association (AMIA) Symposium, 2002*, November 2002.

[24] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating web data. In *VLDB*, 2002.

[25] P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. Ives, F. Pereira, and S. Guha. Learning to create data-integrating queries. In *VLDB*, 2008.

[26] V. Tannen. Provenance for database transformations. In *EDBT*, 2010. Keynote talk; slides available on the author's homepage.

[27] N. E. Taylor and Z. G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD*, 2006.

[28] N. E. Taylor and Z. G. Ives. Reliable storage and querying for collaborative data sharing systems. ICDE, 2010.

# Refining Information Extraction Rules using Data Provenance

Bin Liu[1,*] Laura Chiticariu[2] Vivian Chu[2] H.V. Jagadish[1] Frederick R. Reiss[2]
[1]University of Michigan [2]IBM Research – Almaden

**Abstract**

*Developing high-quality information extraction (IE) rules, or extractors, is an iterative and primarily manual process, extremely time consuming, and error prone. In each iteration, the outputs of the extractor are examined, and the erroneous ones are used to drive the refinement of the extractor in the next iteration. Data provenance explains the origins of an output data, and how it has been transformed through a query. As such, one can expect data provenance to be valuable in understanding and debugging complex IE rules. In this paper we discuss how data provenance can be used beyond understanding and debugging, to automatically refine IE rules. In particular, we overview the main ideas behind a recent provenance-based solution for suggesting a ranked list of refinements to an extractor aimed at increasing its precision, and outline several related directions for future research.*

## 1 Introduction

As vast amounts of data are made available in pure unstructured format on the web and within the enterprise, Information Extraction (IE) – the discipline concerned with extracting structured information from unstructured text – has become increasingly important in recent years. Today, information extraction is an integral part of many applications, including semantic search, business intelligence over unstructured data, and data mashups.

In order to identify regions of text that are of interest, most IE systems make use of *extraction rules*. Rule-based systems such as CIMPLE [10], GATE [8], or SystemT [3, 16], use rules throughout the entire extraction flow. An example rule in these systems might read, in plain English, *"identify a match of a dictionary of salutations followed by a match of a dictionary of last names and mark the entire region as a candidate person"*. The rule would be formally expressed using the system's rule language such as XLog [23] in CIMPLE, JAPE [7] in GATE, or AQL [21] in SystemT. Machine learning-based systems such as [12, 17] use rules in the first stage of an extraction flow, to identify basic features such as capitalized words, or phone numbers. These basic features are fed as input to various machine learning algorithms that implement the rest of the extraction process. We refer the reader to recent tutorials [5, 6, 11] and survey [22] for more details on information extraction systems. Regardless of the kind of system, machine learning-based or rule-based, the output of extraction rules must be extremely accurate in order to minimize the negative impact on subsequent data processing steps, i.e., the rest of the extraction flow, or the application consuming the extracted information.

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

*Work partially done while at IBM Research – Almaden.

Developing highly accurate information extraction rules (or extractors, in short) is a laborious process, extremely time consuming, and error prone. The developer starts by building an initial set of rules, and subsequently proceeds with an iterative process consisting of three steps: (1) execute the rules over a set of test documents, and identify mistakes in the form of wrong results (false positives), as well as missing results (false negatives); (2) analyze the rules to understand the causes of the mistakes; and (3) finally determine refinements that can be made to the extractor to correct the mistakes. This 3-step process is repeated until the developer is satisfied with the accuracy of the extractor.

Traditional research in the field of *data provenance* [2, 24] seeks to explain the presence of a piece of data in the result of a query: *why* it is in the result, *where* it originated in the input database, and *how* it has been transformed by the query. Such kind of provenance has been recently referred to as *provenance for answers*. In contrast, recent work on *provenance for non-answers* [1, 14, 15] seeks to explain the reasons for which an expected result is *missing* from the output. Together, techniques for computing provenance for answers and respectively, non-answers, can be used to explain false positives, and respectively false negatives, therefore facilitating step 2 of the rule development process described above.

While provenance helps the developer in identifying possible causes for incorrect results (step 2), it does not directly address the challenges associated with identifying rule refinements (step 3). In practice, an extractor may contain hundreds of rules, and the interactions between these rules can be very complex. Therefore, designing a rule refinement is a "trial and error" process, in which the developer implements multiple candidate refinements and evaluates them individually in order to understand their effects: *"Are any mistakes being removed ?"*, and side-effects: *"Are any correct results affected ?"* An "ideal" refinement would eliminate as many mistakes as possible, while minimizing effects on existing correct results. Based on our experience building information extraction rules for multiple enterprise software products, designing a single such refinement may take hours. In the same spirit, refining a complex extractor when switching from one application domain to another can take weeks or even months [4, 20].

In this article we explore the question of whether provenance can be used beyond understanding the incorrect behavior of extraction rules, to automatically refine the rules, therefore facilitating step 3 of the rule development process. We start by illustrating the challenges associated with refining extraction rules using a simple motivating example (Section 2). In Section 3, we overview the main ideas behind a recently proposed provenance-based solution for suggesting a ranked list of refinements aimed at improving an extractor's precision that we have implemented in the SystemT[1] information extraction system [3, 16, 21] developed at IBM Research – Almaden. In doing so, we keep the discussion at a high-level, and refer the reader to [19] for technical details. We conclude by outlining several directions for future research in Section 4.

## 2 Challenges in Developing Information Extraction Rules

Figure 1(a) illustrates a simple extractor for identifying mentions of person names consisting of rules $R_1$ to $R_5$. The figure also illustrates the output of the extractor on an input document consisting of a (fictitious) news report as markup in the input text itself, as well as using a familiar tuple-based representation. At a high-level, the semantic of the extraction rules are as follows. Rules $R_1$ and $R_2$ look for mentions of first and last names by identifying matches of entries in dictionaries 'first_names.dict' and 'last_names.dict' in the input text. These mentions are used by rules $R_3$ and $R_4$ to identify candidate person names. In particular, $R_3$ looks for mentions of a first name followed immediately by a last name, therefore marking, for instance, `Morgan Stanley` and `John Stanley` as candidate persons. Rule $R_4$ identifies mentions of last names preceded by a salutation (e.g., `Mr.`, `Ms.`), therefore marking, for example, `Stanley` in "... *Mr. Stanley* ..." as candidate person. Finally, $R_5$ constructs the output of the extractor by unioning together the person mentions identified by $R_3$ and $R_4$.

[1] http://www.alphaworks.ibm.com/tech/systemt

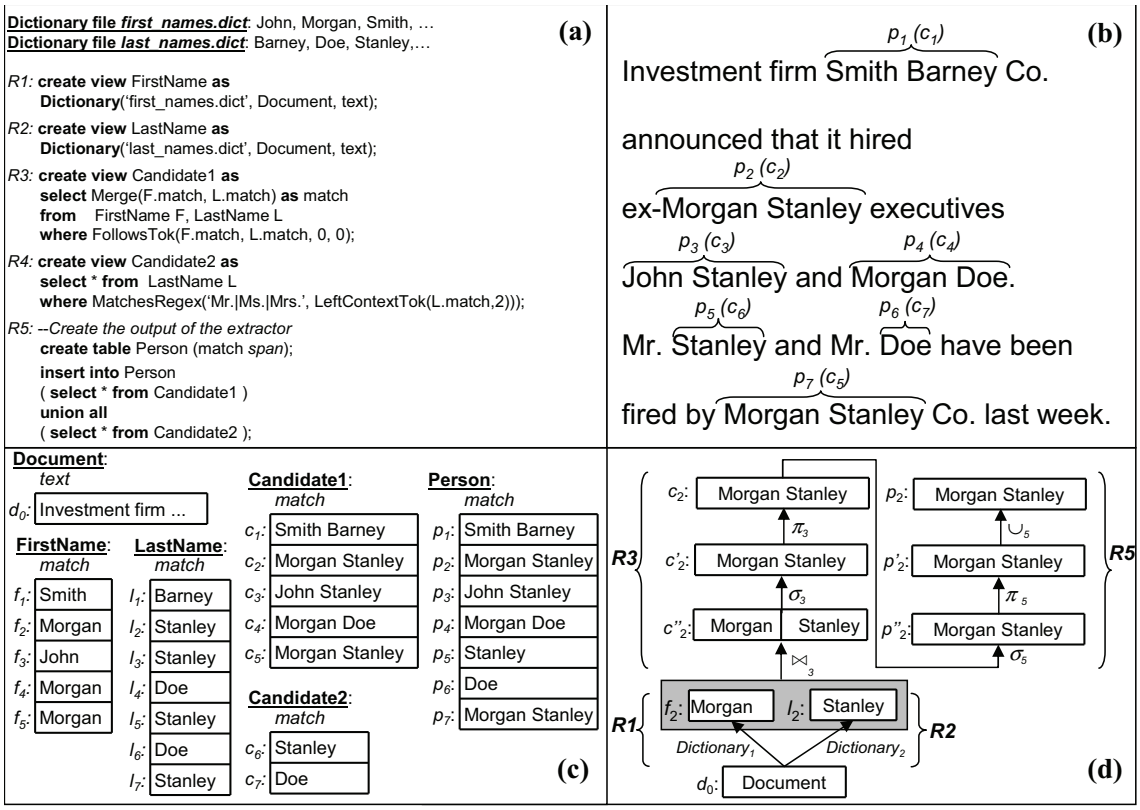



Figure 1: Example Person extractor (a); an input document (b); the results of the extractor on the input document (c); provenance graph of output tuple $p_2$ (d).

As the reader might have gleaned from the example, the basic unit of data in the information extraction context is a region of text, called a *span*. An information extraction program takes as input a document text (represented as relation *Document* with a single span attribute *text*), and outputs a set tuples, each with one or more attributes of type span. A span is associated with begin and end offsets in the input text (see Figure 1(b)). For simplicity, however, we distinguish between spans with the same text value using unique identifiers, see for example $p_2$, and $p_7$. In specifying the extraction rules, we use SQL enriched with a few constructs for expressing text-specific operations, such as basic feature extraction (e.g., *Dictionary()* table function to specify dictionary matching), and span manipulation scalar functions (e.g., combine two spans into a larger span – *Merge()*, select the left context of a span of certain length – *LeftContext()*) and predicates (e.g., check if two spans follow one another within a certain distance – *Follows()*, check that a span satisfies a regular expression – *MatchesRegex()*). The syntax is self-explanatory and we do not insist on the details. We note that we use an SQL-like language for ease of exposition in this article. The language is similar to SystemT's AQL rule language used in our implementation, and can express most popular constructs in other IE rule languages such as JAPE or XLog.

It is easy to see that the extractor in Figure 1 suffers from low precision: while it identifies all correct person mentions, or true positives (e.g., `John Stanley`, Mr. `Stanley`), it also identifies incorrect mentions, or false negatives (e.g., `Smith Barney`, `Morgan Stanley`). Suppose a developer wishes to improve the precision of the extractor. To this end, her goal is to refine the rules in order to remove as many false positives from its output, while preserving true positives as much as possible. Table 1 lists a few of the many possible rule modifications (expressed in both English and our rule language), along with their effects and side-effects. For

| *ID* | *Rule refinement* | *Effect* | *Side-effect* |
|---|---|---|---|
| $M_1$ | Filter Person if it contains the term `Stanley`<br>Add selection predicate *Not(ContainsDict('invalidPersonFragment.dict', match))* to $R_5$ | $p_2, p_7$ | $p_3, p_5$ |
| $M_2$ | Filter Candidate$_1$ if it contains the term `Stanley`<br>Add selection predicate *Not(ContainsDict('invalidPersonFragment.dict', match))* to $R_3$ | $p_2, p_7$ | $p_3$ |
| $M_3$ | Remove entry `Stanley` from dictionary 'last_names.dict' | $p_2, p_7$ | $p_3, p_5$ |
| $M_4$ | Remove entry `Morgan` from dictionary 'first_names.dict' | $p_2, p_7$ | $p_4$ |
| $M_5 (M_5')$ | Filter Person (Candidate$_1$) if it contains the term `Morgan Stanley`<br>Add selection predicate *Not(ContainsDict('invalidPerson.dict', match))* to $R_5$ ($R_3$) | $p_2, p_7$ | – |
| $M_6$ | Filter (Candidate$_1$) if right context contains organization clue, e.g., `Co`<br>Add selection predicate *MatchesDict('orgClue.dict',RightContextTok(L.match)))* to $R_3$ | $p_1, p_7$ | – |
| $M_7$ | Identify person mentions contained within organization mentions;<br>Subtract them from Person | $p_1, p_2, p_7$ | – |

Table 1: Potential rule refinements for the extractor in Figure 1, along with their effects and side effects.

example, $M_1$ adds a predicate to $R_5$ that filters mentions containing the term `Stanley`. While this eliminates two of the three false positives, it also removes two correct results (i.e., `John Stanley`, and Mr. `Stanley`). Alternatively, $M_2$ modifies $R_3$ in a similar fashion with the side-effect of removing a single correct result (`John Stanley`). $M_3$ modifies $R_2$ so that `Stanley` is no longer identified as a last name, which affects both $R_3$ and $R_4$, which will fail to identify correct candidates `John Stanley` and Mr. `Stanley`, respectively. Similarly, $M_4$ modifies $R_1$ so that `Morgan` is no longer identified as first name, thus causing $R_3$ to fail to mark `Morgan Doe` as a candidate person. The rest of modifications do not have any side-effects: $M_5$ and $M_5'$ have the same outcome as $M_2$, while $M_6$ removes `Smith Barney`, but only one of the mentions of `Morgan Stanley`. Finally, assuming the developer has available an extractor for identifying mentions of organizations, she may use it to add a subtraction on top of $R_5$ so that person mentions completely contained within an organization mention are subtracted from the final result. Assuming `Smith Barney` and `Morgan Stanley` are identified as organizations, $M_7$ results in removing all incorrect person mentions, while not affecting any correct ones.

As illustrated by our example, the developer faces several challenges when designing a rule refinement for removing false positives from the result to improve the precision of the extractor.

**Challenge 1: Which rule should be modified ?** In general, there are multiple rules that may be refined in order to remove a false positive. For example, as illustrated by $M_1$ to $M_4$, a modification to any of $R_1$, $R_2$, $R_3$ and $R_5$ may potentially result in removing `Morgan Stanley` from the output. However, the overall effects and side-effects of such potential modifications are very different.

**Challenge 2: What kind of modification should be made ?** Once a rule has been chosen for refinement, there are multiple possible classes of modifications to choose from. For example, should we add a selection predicate as in $M_6$, or a more complex subtraction that involves adding a new rule as in $M_7$ ?

**Challenge 3: How should the modification be parametrized ?** Within each class or rule modifications there are multiple parameters to configure. Consider for example the addition of a selection predicate. Should the predicate be applied to the match itself as in $M_5$, or the context as in $M_6$, and if so, what is the length of the context? Furthermore, should the predicate be based on a regular expression (such as in rule $R_4$), or a dictionary (e.g., $M_5$, $M_6$)? Which regular expression or dictionary is most appropriate?

Although not illustrated by our example, the developer faces similar challenges when designing refinements to address false negatives, with the goal of improving recall. Furthermore, the space of refinements increases dramatically as we move from a simple example to a real-world extractor consisting of hundreds of rules.

## 3 A Provenance Approach to Automatic Rule Refinement

Clearly, it is not feasible for a rule developer to consider all possible refinements. In general, in each iteration of the development process, an experienced developer focuses on a few mistakes that exhibit some commonality. She designs several refinements by considering a few alternatives of rule, type of refinement and parameters, evaluates each refinement individually, and choose one that she finds most appropriate in addressing the target mistakes. Finally, she repeats the process, usually targeting a different class of mistakes. As the amount of human effort required in this "trial and error" process is considerable, an interesting question is whether it is possible to automate some (if not all) of the above steps. In the rest of this article we discuss how provenance enables an approach for systematically and efficiently exploring a large space of possible refinements, and computing a list of "most effective" refinements to be presented to the rule developer.

**Provenance Graph.** While examining the result of an extractor on a collection of documents, the rule developer records information on the results' accuracy. Let us assume that she labels the extracted results as correct (i.e., true positives) or incorrect (i.e., false positives). With this information, we can leverage provenance directly in order to systematically address Challenge 1, i.e., *which rule to modify?* In particular, we are interested in a provenance model that captures *how* an output tuple has been generated by the extraction program from the input document. Such a provenance representation can be obtained by rewriting the extraction program to record, for each tuple generated by the system, the source or intermediate tuples *directly* responsible for generating that tuple. The additional information obtained via the rewritten program can be used to construct a *provenance graph*[2] that illustrates how source tuples and intermediate tuples have been combined by operators of the extraction program in order to generate each of the output tuples. Similar to the representation of SQL using relational algebra operators, we assume a canonical representation of an extraction program as a DAG of operators (e.g., the SPJUD relational operators and text-specific operators such as *Dictionary*). However, in the context of information extraction, none of these operators are duplicate removing, as deduplication is explicitly applied at certain stages of the pipeline using a special consolidation operator. We do not consider this operator in this article, and therefore each output tuple has a single derivation. For example, the portion of the provenance graph corresponding to the derivation of the false positive $p_2$: `Morgan Stanley` is shown in Figure 1(d).

**High-Level Changes.** In order to remove a false positive from the final result, it is sufficient to remove any edge in its provenance graph, since each tuple has a single derivation. Therefore, by examining the provenance of each false positive, we can compile a list consisting of triplets of the form ($Op$, $intermediate_result$, $false_positive$), representing the fact that modifying operator $Op$ in order to remove $intermediate_result$ from its output causes $false_positive$ to disappear from the result of the extractor. We call such triplets *high-level changes*. For example, the following high-level changes would be among those generated for $p_2$: ($Dictionary_1$, $f_2$, $p_2$), ($Dictionary_2$, $l_2$, $p_2$), ($\bowtie_3$, $c_2''$, $p_2$), ($\sigma_3$, $c_2'$, $p_2$), ..., ($\cup_5$, $p_2$, $p_2$). Therefore, the choice of which rule to modify in order to remove $p_2$ from the output is narrowed down to $R_1$ - $R_3$ and $R_5$.

**Low-Level Changes.** A high-level change indicates which rule may be refined, but it does not tell us *how* to refine it (i.e., Challenges 2 and 3). We refer to a specific refinement that implements one or more high-level changes as a *low-level change*. For example, $M_1$ is a possible low-level change for the high-level change ($Dictionary_1$, $f_2$, $p_2$). It is easy to see that any high-level change can be trivially translated into a low-level change that removes exactly one false positive (e.g., $M_5$ and $M_5'$). However, such refinements are not useful in practice, as they do not generalize well to unseen documents. Intuitively, we would prefer non-trivial refinements (e.g., $M_6$ and $M_7$) that remove multiple false positives at a time, and thus have the potential to be effective in general. It turns out that high-level changes are useful towards addressing this problem. Specifically, grouping

[2]The provenance graph can be seen as a graphical representation of an instantiation of *provenance semirings* [13] in which each source (document) tuple is tagged with a unique id, and each derived tuple is tagged with an expression over a semiring having the set of all source tuple ids as domain, and equipped with operations $\cdot$ and $+$ for combining the provenance of tuples involved in a join, and respectively, when a tuple can be derived in multiple ways, and a unary function for representing the operator involved in the derivation.

high-level changes by operator $Op$ produces a maximal set of false positives that can be affected by any refinement of $Op$, along with the specific outputs that should be removed from the output of $Op$ in order to achieve that maximal effect. For example, the three high-level changes for operator $\pi_3$ of $R_3$: ($\pi_3$, $c_1$, $p_1$), ($\pi_3$, $c_2$, $p_2$) and ($\pi_3$, $c_5$, $p_7$) indicate that modifying $\pi_3$ to remove $c_1$, $c_2$ and $c_5$ from its output results in eliminating all false positives $p_1$, $p_2$ and $p_7$. With this information, it is now possible to devise an algorithm for efficiently exploring a large space of types of refinements, as well as algorithms for parametrizing specific types of refinements, while not overfitting to the training document collection.

Given the set of high-level changes grouped by operator $Op$, our algorithm for generating a set of top-$k$ "promising" refinements to be presented to the rule developer proceeds in three steps. First, for each group of high-level changes, we consider different types of low-level changes that can be applied to $Op$. For example, in the case of the $Dictionary$ operator we may consider removing some entries from the dictionary (e.g., as in $M_3$), or changing the matching mode from case insensitive to case sensitive. For the select operator, we may consider adding a selection predicate (e.g., as in $M_6$). In order to be able to evaluate a refinement's quality, the types of refinements considered are carefully selected to guarantee restricting the final result set. Indeed, it is impossible to evaluate a refinement that introduces new tuples in the extraction result, as we have no knowledge of whether these new tuples are correct or not. Second, for each $Op$ and each type of low-level change, we generate $k$ most promising sets of parameters for that low-level change. At the same time, we also compute the effects and side-effects of each concrete low-level change. Finally, we rank the low-level changes generated based on the quality improvement they achieve and present the top-$k$ to the user.

In generating candidate sets of parameters, we apply algorithms specifically designed for each type of low-level change. Before discussing an example of such an algorithm, we briefly overview the main ideas underlying low-level change evaluation, which are shared by all our specific algorithms. A naive evaluation approach would be to simply implement each low-level change and execute the refined extractor. However, this is wasteful and would significantly affect the response time of the system, as the number of refinements considered in a single iteration might be in the order of thousands. A better approach is to traverse the provenance graph and record a mapping between each output tuple and each of the intermediate tuples involved in its provenance. By reverting this mapping we can obtain, for each intermediate tuple $t$ in the result of some operator $Op$, the set of output tuples depending on it. Since each tuple has a single derivation tree, modifying $Op$ to remove a tuple $t$ from its local output guarantees the removal of all tuples depending on $t$ from the final result. Furthermore, since each refinement can only restrict the final result, we also know the labels of all these tuples. Therefore, we can determine the effects and side-effects of the refinement on overall result quality by unioning this information for all tuples removed by the refinement from the local output of $Op$.

A detailed discussion of specific algorithms for parametrizing low-level changes is outside the scope of this article. To give the reader a flavor, however, we briefly describe the algorithm for parametrizing filter selection predicates (as in $M_6$) next. The algorithm systematically considers the tokens to the left or right of each span in a tuple affected by a high-level change. For each choice of left/right context of a given length up to a preset maximum value, the union of the contextual tokens forms a set of potential dictionary entries. For example, in the case of operator $\sigma_3$ of $R_3$, the tokens `Co` and `executives` would be considered as potential dictionary entries for a filtering predicate on the right context of length 1. We now need to evaluate the effects of using various subsets of these dictionary entries to filter the local output of $\sigma_3$. However, the number of all subsets can be prohibitively large. To avoid evaluating all of them, we adopt a greedy approach: we group together tuples according to which dictionary entries occur in the vicinity of their spans, evaluate the effect of each entry on result quality using the approach described above, and generate $k$ refinements, each using top-$k$ most effective entries ($k$ is the maximum number of refinements that will be presented to the developer). Two refinements would be generated for our example, one using `Co`, while the other using both `Co` and `executives` as filter.

## 4 Conclusion and Directions for Future Research

We discussed some of the challenges in designing high-quality information extraction rules, and showed that provenance plays an important role in facilitating rule development that goes far beyond the general consensus that *"provenance is useful in understanding the program and its output"*. In particular, we discussed the main ideas behind a framework for automatically generating rule refinements aimed at improving the precision of an extractor based on user labeled examples. Very recently, the idea of extractor refinement based on provenance and labeled examples has been also approached in [9] in the context of iterative IE systems, however, refinements are considered only for the last step of an extraction flow, and not an entire extraction program as we do here.

Our approach is only a first step towards a general framework for automatic refinement of information extraction rules. We conclude by outlining several possible extensions.

**Precision-driven Refinement.** Besides the *provenance* $\rightarrow$ *high-level change* $\rightarrow$ *low-level change* paradigm, the usefulness of the framework lies in efficient algorithms for generating concrete refinements. While the framework implements several such algorithms, more could be added. In particular, incorporating existing approaches for learning basic features such as regular expressions [18] would be extremely useful. Furthermore, the assumption made in Section 3 is that an output tuple has a single derivation. Therefore, removing any single intermediate result involved in its provenance is sufficient to eliminate that output tuple. Formalizing the notion of high-level change in the case of multiple derivations, and designing efficient algorithms for generating concrete refinements in this context are interesting directions for future work.

**Recall-driven Refinement.** Our current framework handles refinements geared towards improving precision. However, equally important is improving the recall, i.e., generating refinements that remove false negatives, while not affecting existing correct results. Towards this end, it would be interesting to extend our framework to incorporate recent techniques for computing provenance of missing answers [1, 14, 15]. The notion of high-level change, and algorithms for efficiently generating low-level changes need to be revisited in this context.

**Active Learning of Labeled Examples.** The ability to automatically generate useful refinements depends on the number, variety and accuracy of the labeled examples provided to the system. While labeled data for certain types of popular extractors on general-purpose domains is publicly available [26], in most cases the labeled data must be provided by the rule developer. Unfortunately, labeling data is itself a tedious, time-consuming and error prone process. It would be interesting to investigate whether active learning techniques [25] can be used to present to the developer only those most informative examples, therefore facilitating the labeling process.

## References


[1] A. Chapman and H. V. Jagadish. Why Not ? In *SIGMOD*, pages 523–534, 2009.

[2] J. Cheney, L. Chiticariu, and W. Tan. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[3] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. SystemT: An Algebraic Approach to Declarative Information Extraction. In *ACL*, 2010.

[4] L. Chiticariu, R. Krishnamurthy, Y. Li, F. Reiss, and S. Vaithyanathan. Domain Adaptation of Rule-based Annotators for Named-Entity Recognition Tasks. In *EMNLP (To appear)*, 2010.

[5] L. Chiticariu, Y. Li, S. Raghavan, and F. R. Reiss. Enterprise Information Extraction: Recent Developments and Open Challenges. In *SIGMOD (Tutorial)*, pages 1257–1258, 2010.

[6] W. Cohen and A. McCallum. Information Extraction and Integration: An Overview. *KDD(Tutorial)*, 2003.

[7] H. Cunningham. JAPE: a Java Annotation Patterns Engine. Research Memorandum CS – 99 – 06, University of Sheffield, May 1999.

[8] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *ACL*, 2002.

[9] A. Das Sarma, A. Jain, and D. Srivastava. I4E: Interactive Investigation of Iterative Information Extraction. In *SIGMOD*, pages 795–806, 2010.

[10] A. Doan, J. F. Naughton, R. Ramakrishnan, A. Baid, X. Chai, F. Chen, T. Chen, E. Chu, P. DeRose, B. Gao, C. Gokhale, J. Huang, W. Shen, and B.-Q. Vuong. Information Extraction Challenges in Managing Unstructured Data. *SIGMOD Record*, 37(4):14–20, 2008.

[11] A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Managing Information Extraction: State of the Art and Research Directions. In *SIGMOD (Tutorial)*, pages 799–800, 2006.

[12] D. Freitag. Multistrategy learning for information extraction. In *ICML*, 1998.

[13] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance Semirings. In *PODS*, pages 31–40, 2007.

[14] M. Herschel and M. Hernandez. Explaining Missing Answers to SPJUA Queries. *PVLDB (to appear)*, 2010.

[15] J. Huang, T. Chen, A. Doan, and J. F. Naughton. On the provenance of non-answers to queries over extracted data. *PVLDB*, 1(1):736–747, 2008.

[16] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. SystemT: a System for Declarative Information Extraction. *SIGMOD Record*, 37(4):7–13, 2008.

[17] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[18] Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. V. Jagadish. Regular expression learning for information extraction. In *EMNLP*, pages 21–30, 2008.

[19] B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. R. Reiss. Automatic Rule Refinement for Information Extraction. *PVLDB (to appear)*, 2010.

[20] D. Maynard, K. Bontcheva, and H. Cunningham. Towards a semantic extraction of Named Entities. In *RANLP*, 2003.

[21] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. An Algebraic Approach to Rule-Based Information Extraction. In *ICDE*, pages 933–942, 2008.

[22] S. Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[23] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*, pages 1033–1044, 2007.

[24] W. C. Tan. Provenance in Databases: Past, Current, and Future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.

[25] C. Thompson, M. Califf, and R. Mooney. Active Learning for Natural Language Parsing and Information Extraction. In *ICML*, pages 406–414, 1999.

[26] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In *CoNLL at HLT-NAACL*, pages 142–147, 2003.

# Socializing Data with Google Fusion Tables

Hector Gonzalez Alon Halevy Anno Langen Jayant Madhavan Rod McChesney
Rebecca Shapley Warren Shen Jonathan Goldberg-Kidon

**Abstract**

*We describe the social features of Google Fusion Tables, a cloud-based data management service whose goal is to facilitate collaboration around data sets. The social features include the ability to specify attribution of data sets, a mechanism for conducting discussions on data (at fine granularity, such as row, column or cell), the ability to merge tables that belong to different owners, and the ability to share specific queries and visualizations and embed them in other properties on the Web. We describe the rationale for designing these features and our experiences after our first year of interacting with users.*

## 1 Introduction

Google Fusion Tables [7, 8] is a cloud-based service for data management and integration. The over-arching goal of Fusion Tables is to facilitate new types of collaboration around private and public data sets and to enable data management by a broader audience of users. Though we have witnessed a wide range of applications of Fusion Tables, we have seen considerable usage by organizations that are struggling with making their data available internally and externally, and communities of users that need to collaborate on data management across multiple enterprises.

Fusion Tables enables users to upload tabular data files (spreadsheets or CSV) of up to 100MB. The system provides several ways of visualizing the data (charts, maps, time-lines), and the ability to query by filtering and aggregating the data. We support integrating data from multiple sources by performing joins across tables that may belong to different users. Derived tables can be created by joining multiple tables or by projecting a subset of the columns from a single table. Users can keep the data private, share it with a select set of collaborators, or make it public. Fusion Tables also supports an API which many have used to build interesting applications.

As we gathered requirements for Fusion Tables it was clear that provenance and lineage need to be attached to tables in such an open environment. However, we quickly realized that there is a much broader set of features that are needed to support rich social interactions. This paper focuses on the social features of Fusion Tables:

- Attribution: when a user uploads a table into Fusion Tables, she is given the option to attach an attribution to the data. The attribution is the mechanism for specifying the provenance of the data. The attribution is attached to any visualization or table that is derived from the table. Attribution is crucial for two reasons. First, it is important for users to know the source of the data they are looking before they can trust it or use combine it in interesting ways. Second, many owners of data are reluctant to share data unless the attribution is attached. Hence, attribution plays a key role to entice organizations to share their data.

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

- Discussion: Fusion Tables enables users to discuss the data at a fine granularity: individual rows, columns and cells. Discussions are important in order to enable users to improve the quality of the data, clarify the assumptions underlying it, or opine on its meaning.
- Data integration: as users find tables produced by others, they want to combine them to create new tables. Fusion Tables enables a user to merge any pair of tables as long as the user has read permissions on it. The system also enables users to inspect the lineage of derived tables.
- Publishing: Fusion Tables offers several easy ways of publishing data. A user can create a saved link of a table or a visualization that can be shared via social media. Users can also embed a visualization in other web pages by cutting and pasting an HTML snippet. Publishing data effectively is an important component of enticing people to create interesting data sets.

This paper discusses the social aspects of fusion tables and the rationale underlying their design (Section 2). We then discuss some of the experiences we have had with these features and the challenges that lie ahead 3).

## 2 Social features of Fusion Tables

We designed the social features of Fusion Tables to facilitate collaboration among multiple parties, the ability to enhance the data, create new data sets, and to provide easy discovery and dissemination of data. In what follows we describe the features of Fusion Tables that are focused on social interactions.

**Attribution:** In an open data environment such as the Web, it is crucial that data owners get credit for publishing their data and that users know the provenance of the data as they inspect it or use it further. As we were building Fusion Tables, the need to specify attribution came up as a key requirement.

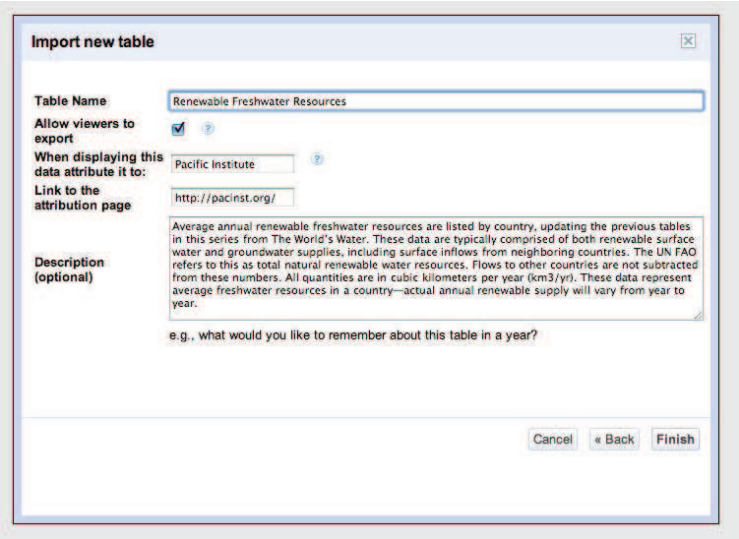



Figure 1: Specifying attribution in Fusion Tables. Users are encouraged to provide the attribution as a string and a URL. They are also encouraged to describe their data during the import process. In addition, users can specify whether the data can be exported into CSV files by other parties.

Users specify the attribution of a table in initial import flow (see Figure 1). Our import flow is designed to have as few steps as possible in order to make import simple. Even so, we felt that specifying attribution

is important enough to be included. The attribution includes both a string value and a URL of the source. It is important to note that you do not need to be the originator of the data in order to give the attribution. For example, in Figure 1 a user is importing data that he found (and attributes to) the Pacific Institute. On the Web it is common for users to upload data that is attributed to other organizations.

The attribution is always shown along with the data, whether it is visualized as a table or using a chart or a map. As the data gets processed, the attribution is propagated appropriately.[1] When two tables are joined, the attributions of both tables are shown and the interface on the table view shows the columns from the different sources in slightly different colors. The attribution is also propagated to views and selections of the data. Given the current query capabilities of Fusion Tables, there are no subtle issues related to propagating attribution – the attribution always includes the attributions of all the base tables involved in a view. Users can also inspect how a table is derived from other base tables in the About menu. We note that lineage on views is shown at the level of tables. We do not currently show how each row in a derived table is derived from rows in the base tables.

**Discussion:** Viewing and querying data are just a first step in a real collaborative effort. In many applications where data is shared among multiple parties, there needs to be significant discussion about the data. The discussion can lead to correcting data values, expressing additional opinions about the data, and elucidating the assumptions underlying the data.

Given that we aim to support large collections of data, it was necessary that we enable users to discuss data at varying levels of granularity – discussing 1 million rows in one block of text would be an ineffective way of keep track of conversations. Hence, Fusion Tables enables conducting discussions at different levels of granularity, including individual rows, columns and cells (see Figure 2).

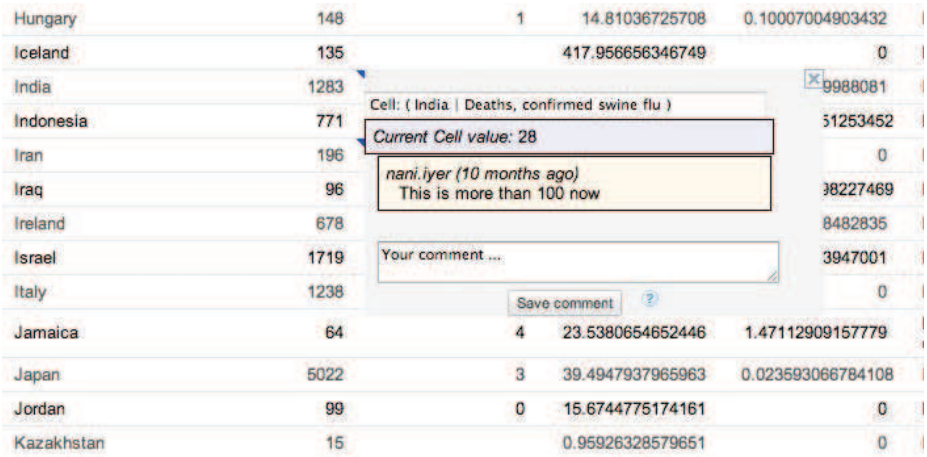



Figure 2: Discussion on cells in Fusion Tables. Discussions can be conducted on individual rows, columns or cells. Discussions are shown as trails, and if the value of the data is changed, then the change is shown as part of the trail to provide context.

Discussions are presented as a record of statements through time, similar to instant messaging interfaces. When a value in a cell changes, the change is shown as part of the discussion trail, and therefore the context

[1]When data is embedded in other web sites, we currently do not show the attribution, mostly because we have not yet found an appropriate place in the UI.

of the change is visible. Fusion Tables also provides a discussion panel, where the user can search through a collection of discussions, rather than having to view each one.[2]

Unlike attribution, we do *not* propagate discussions on a table to tables that are derived from it. The rationale for this design is that a view may be used for a different purpose. For example, you may want to conduct two parallel discussions on a different views of the same underlying data with two distinct set of collaborators. There are cases where users would like to propagate discussions to views, and therefore in, in principle, we could leave the propagation decision to the user (using the mechanisms developed in DBNotes [5]).

One of the emergent use cases of the discussion feature has been to annotate the data with additional explanation. For example, a column may use a convention to describe different health facilities in a country, and a comment on the column can explain the meaning of the different conventions. While the same information could have been put in the text description of the table, putting it in a comment makes it more accessible while viewing the data. However, explanations of the data are likely valuable to all users of the data, and less likely to need to be kept limited to a private set of collaborators. This is an example of where the capability to over-ride the decision not to propagate discussions to views would be useful.

**Data integration:** The ability to find tables produced by others raises the potential of combining multiple tables. For example, if you own a table about coffee production by different countries in the world, you may want to combine it with tables containing the GDP or population of these countries. More generally, combined data sets enable users to discover deeper correlations or trends. As shown in Figure 3, Fusion Tables currently lets users merge (i.e., join) two tables based on a shared column. When two tables are combined, the permissions on the individual columns are the same as for the base tables.

Some subtle issues arise as we propagate permissions on tables that are derived by merging. Suppose user $A$ wants to grant permissions on table $T$ to user $B$. If the table $T$ is the result of fusion tables $T_1$ and $T_2$, then $A$ may not have write permissions on all of $T$'s columns. Hence, $A$ can only grant write permissions to $B$ on columns of $T$ for which $A$ himself has write permissions. We note that join column used to create a merged table is never directly editable on the merged table itself. All changes to the merge column need to be made on the base tables.

**Publishing data:** There are several mechanisms for sharing data in Fusion Tables. For starters, an owner of a table can share it with a set of collaborators by specifying their email addresses. The collaborators can be granted read and/or write permissions on the table. The owner can also specify *merge* permissions, where the collaborator can add new columns to the table but cannot change the data in the existing columns. Merge permissions are useful in cases where multiple parties are creating a data set together, but each one has authority over a particular set of columns.

Tables can also be made public to be viewed by anyone. The table owner can decide whether to allow search engines to crawl the table or not. If not, the data may be shared by sending around a link. In addition to showing up in general search results, Fusion Tables provides a search facility over the tables that have been made public and crawlable in the spirit of [6].

We quickly found out that users want share visualizations of the data in addition to the data itself. The reason is that creating a visualization (which may apply to a carefully chosen subset of the table) can convey a particular point and involve more intimate knowledge of the data than can be expected by the viewers. Hence, Fusion Tables enables users to create *saved links* of the data. A saved link is a visualization (e.g., map, bar chart) applied to some query over the data. The link can be shared using any social media (e.g., Twitter or Facebook).

Visualizations of the data can be embedded in other Web properties. For example, as illustrated in Figure 4, a common use case of Fusion Tables is by journalists who want to back up their article by relevant data. Such

[2] At the storage level, comments are stored in a separate table than the data itself. We store all the comments for all user tables in a single Bigtable table. The key of the comments table is the subject of the com- ment, which is the triple: (table, row, column). It uniquely identies the element a comment applies to. The value of a row is the text of the comment, the author, and the date the comment was posted.

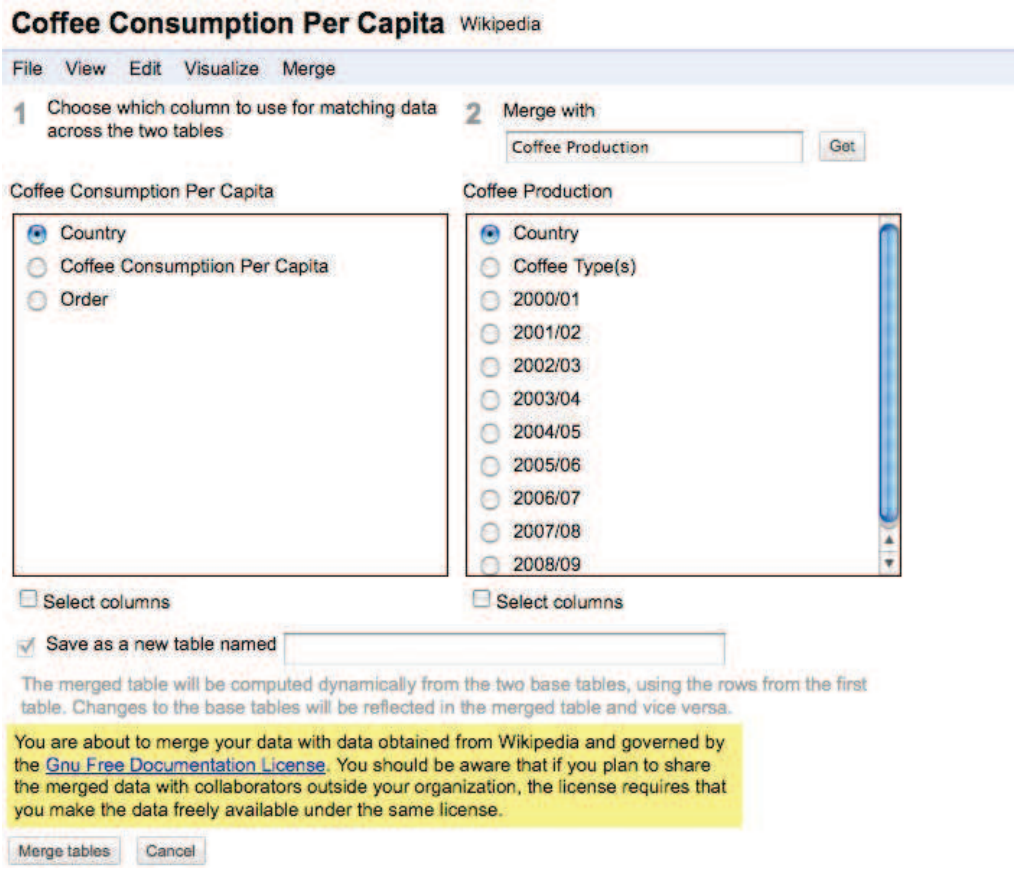



Figure 3: Integrating two tables via a merge operation. The user can search for tables to merge with, and then specifies the join column in both tables. In the figure, the user is merging the table Coffee Consumption per Capita with the table Coffee Production on the Country' column.

embeddings are a significant source of our traffic, because the data is served to users who are browsing other popular Web pages. The owner of the data can see how many views a data set received, which is another mechanism to create an incentive for owners to share their data.

## 3 Experiences and Challenges

We learned several lessons from our experience with Fusion Tables and from users' requests. These lessons have given rise to some challenges in our system and lead to some interesting future work. We summarize the main ideas below.

**Attribution and provenance:** We observed that 12% of the tables in the system provide some attribution, and among tables that have been made public 34% have attribution. One of the issues that was raised by our users is the desire to distinguish between the case in which the data is uploaded by the actual owner/producer of the data versus the case in which someone uploaded data they found and want to provide the appropriate attribution. Of course, in the current state attribution can be abused, but we have not witnessed that happening yet. Currently, the only way to check whether the data was uploaded by the owner is if their email address matches the appropriate organization.

Figure 4: Embedding a map visualization from Fusion Tables in a news article. Updates to the underlying table will be automatically propagated to the map visualization.

Given our current experience, we view the attribution feature of Fusion Tables as reasonably effective. We note that comments in discussions also contain the email address of the person commenting. Currently, we do not let users query for comments by owner, though that could be a useful feature.

We see attribution and provenance playing a significant role in socializing data. In particular, there are domains in which the source of the data and the exact method that derived data was computed are crucial. For example, in climate science or bio-diversity many of the discussions among scientists and the messaging of science to the general public is based on derived data sets. But given the diversity of opinions in these fields and the fact that the science is constantly evolving, it must be possible to specify where the data came from and when and how it was derived from other data. Such provenance is necessary in order to ensure that the assumptions underlying the data are appropriate for the implied conclusion, and in order for other scientists to be able to recreate the same results.

**Discussion:** The experience with the discussion feature has been much more mixed and we believe there are many more challenges there. Our main observation from talking to users is that the role of the discussion feature in collaboration is not yet clear. Specifically, users do not always understand the purpose of leaving a comment and what effect it will have. For example, they do not know whether someone will actually reply or act on a comment that was left on a cell or row. As a result, they resort to the usual ways of discussing data (e.g., email

and chat).

One strategy to address the confusion issue is to provide discussion mechanisms with more specific semantics. For example, we can enable a data owner to set up a comment where she solicits ratings for particular rows. As another example, we can specify that a comment is an explicit proposal to change or correct a value in a cell. The challenge here would be to select a relatively small set of discussion types that would satisfy users' most common needs.

Another feature requested by our users is to be able to provide a comment on a *set* of cells (which cannot be described as a query in a simple fashion). For example, such a comment would specify that a set of values seem inconsistent, or that the cells together exhibit an interesting phenomenon that deserves attention.

Users have also noted the desire to discuss specific visualizations or parts of a visualization. For example, when viewing a scatter plot of a data set you may notice a set of outliers, and it is much more effective to mark them on the visualization itself. Annotating on visualizations raises the challenge of propagating the annotation to the underlying data (and from there to other visualizations, when relevant). Basic functionality for annotating visualizations is supported in ManyEyes [1].

Discussions often refer to specific versions of the data. Fusion Tables does not currently support creating snapshots of the data and referring to them later. Several users have noted that adding snapshots would increase the variety of cases in which our discussion features could be used.

**Data integration:** Our data integration features are quite basic at this point, and this was an explicit decision when we designed Fusion Tables. The goal was simply to let users see different data sets side by side, without having to perform any schema-level integration. One of the common requests is to allow more flexibility in joining two tables. In particular, the rules for matching pairs of cells from the two join columns should be more flexible. Typically, users merge a pair of tables, but we have seen cases of up to up to eleven base tables being merged[3]

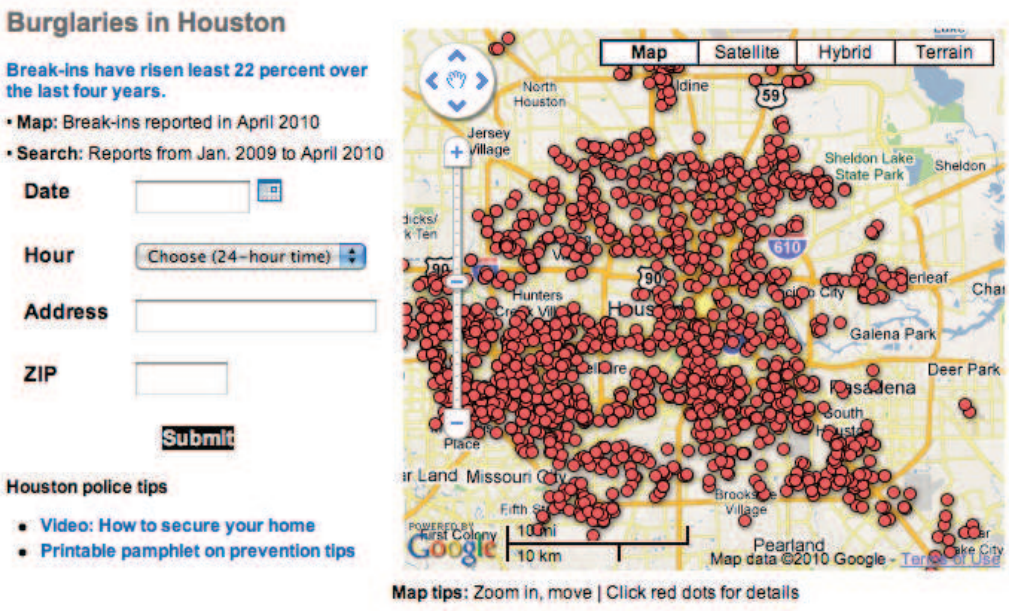



Figure 5: A page on the Houston Chronicle that crowd-sources data about burglaries in the Houston area. Readers are asked to report known burglaries on the accompanying web form.

[3]See http://tables.googlelabs.com/DataSource?dsrcid=183101.

**Crowd sourcing:** One of the interesting applications enabled by Fusion Tables is the ability to crowd-source data that would otherwise be hard to obtain. For example, Figure 5 shows how the Houston Chronicle asks their readers to report burglaries in the Houston area. We have seen several cases where Fusion Tables has been used for this purpose. Crowd-sourcing raises several issues that we will need to address in Fusion Tables, such as being able to determine which data is reliable, which is accurate and up to date and which is redundant. Currently all of these issues need to be addressed in the code that implements the crowd-sourcing.

# 4 Conclusion

Recently, there have been many calls for an *open data movement*, where data is made public by organizations, curated and enhanced by people who care about it, and then disseminated to a much wider audience. Such open data projects have the potential of affecting decision making throughout many organizations, and making high-quality data more readily available to those who need it.

Fusion Tables and related products such as Factual [2], Socrata [3] and Swivel [4] aim to support the interactions needed for open data using a set of social mechanisms to encourage collaboration. This article discussed the social features we currently support in Fusion Tables, but we note that the area of supporting social aspects of data management is still very much in its infancy. We described some of the challenges we have gleaned from our users, but we are sure more challenges will soon become evident.

# References


[1] www.manyeyes.com.

[2] www.factual.com.

[3] www.socrata.com.

[4] www.swivel.com.

[5] D. Bhagwat, L. Chiticariu, W. Tan, and G. Vijayvargiya. An Annotation Management System for Relational Databases, In *VLDB Journal*, 2005.

[6] 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.

[7] H. Gonzalez, A. Halevy, C. Jensen, A. Langen, J. Madhavan, R. Shapley, and W. Shen. Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. In *SOCC*, 2010.

[8] 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.

# A Rule-Based Citation System for Structured and Evolving Datasets

Peter Buneman
University of Edinburgh
opb@inf.ed.ac.uk

Gianmaria Silvello*
University of Padua
silvello@dei.unipd.it

**Abstract**

*We consider the requirements that a citation system must fulfill in order to cite structured and evolving data sets. Such a system must take into account variable granularity, context and the temporal dimension. We look at two examples and discuss the possible forms of citation to these data sets. We also describe a rule-based system that generates citations which fulfill these requirements.*

## 1 Motivation

Citations are one of the most significant tools used in the creation and propagation of knowledge; through citations we can give credit or attribution to researchers, convey a brief indication of the contents (such as a title), and provide location information for retrieval of the referenced work. Over the years, a well-developed system (with minor variations) has evolved for the citation of sources such as books, journals and scholarly papers. On the other hand, there is no well-defined mechanism for publishing, accessing and citing datasets [9, 1, 8]. Datasets are difficult to catalog [8] and, consequently, they are difficult to cite, retrieve and reuse. We should note that publishing datasets that are subsequently cited is beneficial for scientists, who may receive credit for their work of collecting and assuring quality of the dataset and for organizations such as digital libraries or funding bodies that want to build accessible data collections for the benefit of designated communities [6].

We start with the observation that a citation is more than a persistent identifier such as a URI or *Digital Object Identifier (DOI)*[1] which allow us to *locate* the data of interest; citations also carry certain kinds of metadata such as a title, authorship, date – information that is immediately useful in judging the quality or currency of the cited information. Moreover, citations are increasingly used to judge the reputation of a work or person. We believe that these functions of traditional citation should carry over to data citation, but in order to pursue this idea we need to make the following assumptions: (a) that there is an accepted system of persistent identifiers, (b) that the referenced material does not change and (c) that it is hierarchical in structure. All three of these require some consideration. For (a) there is still no universally accepted system. Many data sets change in time, and old versions are not always available in order to satisfy (b). While (c) is required for the mechanisms we propose

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

*Work of this author was performed at the University of Edinburgh, while visiting from the University of Padua.
[1] http://www.doi.org/ Note that this is neither a persistent identifier nor a citation.

and is usually true for scientific data sets, it is not clear how it applies to ontologies, something that we briefly discuss at the end of this paper.

In a conventional citation, information such as ISBN, journal number, page number etc. serve uniquely to identify a work and also may provide some help in retrieving the work (e.g. once you are inside the library.) A persistent identifier such as the DOI provides unique identification and may also be associated with a means of retrieval such as a URL. In order to fulfill the requirements of a conventional citation we clearly have to associate further information with the DOI. While it is common practice to do this for digital objects, there are no conventions or standards and, most importantly for data citation, we are typically citing a part of some larger collection. Here neither the location information within the collection nor the metadata associated with the cited component can be carried with the DOI. In fact, we believe the situation is the reverse. The DOI should be a part of the citation.

We consider the citation issues related to the field of Digital Libraries and of Curated Databases. In the first case we consider how to cite digital archives described by means of the international standard for archival description which is the *Encoded Archival Description (EAD)*[2]. For the latter case we consider a widely used curated database which is the IUPHAR DB[3] incorporating detailed pharmacological, functional and pathophysiological information.

In this paper we present the requirements that a citation system must fulfill in order to guarantee the consistency and the integrity of citations. Furthermore, we take into account the concept of *citable unit*, explaining how it is adopted in the citation practice of traditional publications and how it can be exploited in the citation of structured and evolving datasets. We show how this concept is implicitly adopted by different citation systems based on the use of unique identifiers. We present the advantages but also some issues of these citation systems; in particular we put in relation the EAD files and the IUPHAR DB with these citation systems pointing out where a new solution is necessary to overcome some citation issues. The final section of this paper presents the rule-based citation system that we have designed and developed[4], showing how it addresses the presented issues and how it can be further extended.

## 2 Two examples: EAD and the IUPHAR Database

Of our examples, the EAD represents a widely-adopted standard to represent and describe archival resources; IUPHAR is a curated database that resembles a conventional publications such as a reference manual in that it represents the work of a large number of experts who both create and revise its content. These two datasets are quite different in their nature and use, but they both have a hierarchical structure that can be exploited for the definition of an automatic citation system. Furthermore, they represent two different means of representing and disseminating the information which has the same citation issues and that can benefit from the solution we propose.

In order to understand the characteristics of the **EAD** file we need to introduce some basic concepts about archives. An archive is a complex organization which is not just a series of (digital) objects that have been accumulated over time. Archives have to keep the context in which their documents have been created and the network of relationships among them in order to preserve their informative content and provide understandable and useful information over time. This is achieved through a richly annotated hierarchical classification system.

In a digital archive the components are described by the use of metadata that can express and maintain this structure and relationships. The standard format of metadata for representing the complex hierarchical structure of the archive is EAD [11], which reflects the archival structure and relationships between documents in the archive. To maintain all this information an EAD file turns out to be a large *eXtensible Markup Language (XML)*

[2] `http://www.loc.gov/ead/`

[3] `http://www.iuphar-db.org/`

[4] The Java API can be found here: `http://www.dei.unipd.it/~silvello/datacitation/`

file with a deep hierarchical internal structure. In Figure 1 we can see how the XML structure of an EAD file resembles the structure of the archive[5].

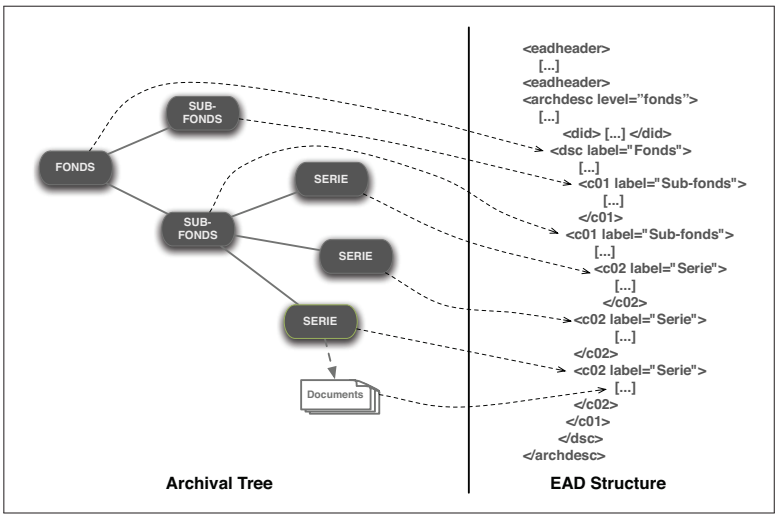



Figure 1: The hierarchical structure of an archive mapped into an EAD file.

We can see that every node in the archival hierarchy is mapped in a specific tag of the EAD XML file, where: The `<eadheader>` contains metadata about the archive descriptions and includes information about them such as title, author and date of creation. The `<archdesc>` contains the archival description itself and constitutes the core of EAD; it may include many high-level sub-elements, most of which are repeatable. The most important element is the `<did>` or descriptive identification, which describes the collection as a whole; it is composed of numerous sub-elements that are intended for brief, clearly designated statements of information and they are available at every level of description. Finally, the `<archdesc>` contains an element that facilitates a detailed analysis of the components of a fond, the `<dsc>` or description subordinate components. The `<dsc>` contains a repeatable recursive element, called `<c>` or component. A component may be an easily recognizable archival entity such as series, subseries or items. Components are not only nested under the `<archdesc>` element, they usually are nested inside one another and they are indicated with `<c...>` tag.

**The IUPHAR database** incorporates detailed pharmacological, functional and pathophysiological information on G Protein-Coupled Receptors, Voltage-Gated and Ligand-Gated Ion Channels. Although the database is internally represented as a relational database, its structure as see by both the users and the contributors is essentially hierarchical, where the root is the database as a whole and it has the list of receptor families as children nodes; each receptor family node has the receptors as children nodes. Each receptor is described by a Web page containing the main technical information; these Web pages contain information such as lists of "agonists", "antagonists" for each receptor.

## 3 Requirements of a Citation System

In order to define the requirements that a citation system for structured and evolving datasets has to satisfy, we need to identify which are the "citable" elements in a dataset. To accomplish this purpose we have to introduce an important concept, i.e. the "citable unit". The concept of citable unit has been used in general terms to indicate the target of a citation such as a book, a chapter, a journal or a paper [10]. In [2] the concept of citable

[5]For the uninitiated a *fond* is a term derived from French that describes the collection of records in an archive that originate from the same creator. Please see: `http://www.archivists.org/glossary/term_details.asp?DefinitionKey=196`

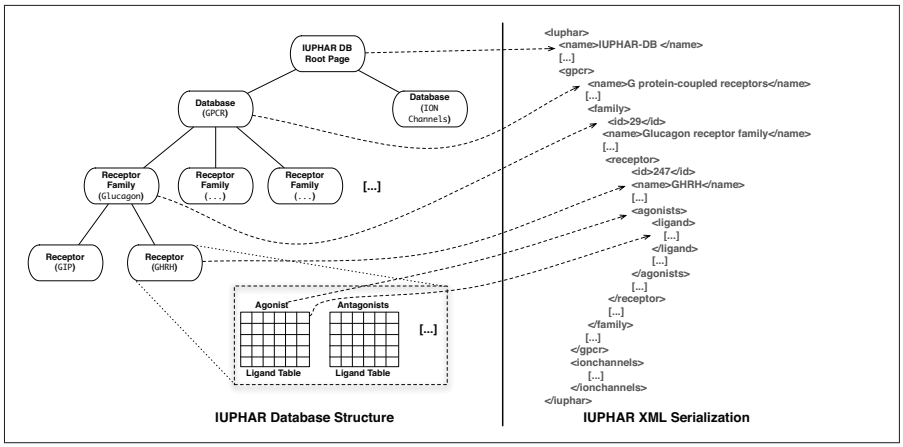



Figure 2: The structure of the IUPHAR Database and its XML serialization.

unit is exploited in the law documentation context to indicate each element of an "Act of Parliament" that must be unambiguously identified. While the object of interest may be a simple fact or a single number, the citable unit is the context in which that object occurs. Note that what constitutes a citable unit is usually understood: a book or an article, but there are cases in which it may not be clear. For example, in a book which is a tightly integrated collection of chapters by different authors, is the citable unit the book or the chapter; or should both be allowed?

For the hierarchical structures we are considering we believe that it should be possible to cite nodes at various levels in the database. For example, in IUPHAR, the citation for "The IUPHAR database does not contain $X$" is the whole database, however for "$Y$ is an agonist for $X$ it is the section on receptor $X$ (the authorship for each family of receptors is different). Thus it should be possible to have one citable unit contained in another. Furthermore, we believe that the onus is on the *publishers* of the data to define the citable units.

Most of the data citation proposals [9, 1, 8] are what we call "identifier-based" – they start with the idea of assigning a unique identifier (e.g. a DOI) to the objects that need to be independently cited by and by associating appropriate metadata for that object. The main differences between these proposals are in the forms of metadata to be associated with the identifier. These solutions implicitly adopt the concept of citable unit as a foundational idea; indeed, they identify each citable unit with a unique identifier and then use the associated metadata for citations. They appear not to take into account the hierarchical nature of the objects being cited. It should also be noted that citable units can be very large (a whole book or database) and that it is often useful to have a means for locating a given "small" element of information – a fact or figure – within that unit. Thus some location system may be needed for this, but as we shall see, this does not necessitate an independent global digital identifier for each such element.

Let us see how these *identifier-based* solutions behave with the presented datasets; we can point out two main requirements that a citation system has to support: deep citations and temporal invariance. Deep citations can be divided into two parts: variable granularity and context. Each section in an EAD file may be identified by means of a unique identifier with associated metadata. At the same time we may want to cite a particular document contained in a node of the archive (e.g. a document contained in a series); in this case we need to have an identifier and a metadata for each document described in the EAD file and not only for the main parts representing the archival divisions. If we consider the IUPHAR database, as we have just seen there are good reasons for making nodes at different levels available as citable units. Moreover the metadata associated with each unit will change: the "authorship" of the whole database is different from that of its sub-units; the "title" will be different; and the currency (the date of last modification) may also differ.

Even though the number of citable units in a hierarchy may be limited, we may want – in a citation –

to provide information about a node that is not a citable unit, but is part of one. Typically this is location information for that node *within* the citable unit. When a citation is used in the form "In $C$ it is stated that $X$", it is often useful to have some further information, e.g., "page 305", about where in $X$ is located in $C$; but we do not want to use $X$ (or its location) itself as the citable unit. This is would be use out of context. The same is true of the data sets we have been discussing: a single data value is not itself a citable unit, but having its location within a citable unit can be enormously useful for verification.

The second requirement is that *once a dataset has been cited, it must remain stable and always accessible in the cited form*; this involves the **temporal issues** connected with evolving datasets such as curated databases. In order to address this issue a reasonable solution is to employ a versioning system that permits us to access a specific version of a dataset or of a component of a dataset. In any case, we have to take into account that the adoption of a versioning system is not a one-size-fits-all solution; for instance, with highly dynamic datasets the volume of changes can be so large or frequent to make tracking back difficult to manage [8]. The temporal requirement is particularly relevant in the case of the IUPHAR DB because it changes quite often; indeed, the information about a receptor may be updated or new families can be added, some errors can be corrected etc. In the case of EAD files this is less frequent because archival information is relatively stable. For this reason we present our citation system starting from its use with the EAD files and then with the IUPHAR database which requires additional work to deal with the evolution of the data set.

# 4 A Rule-Based Citation System

We now describe a simple rule-based system for the automatic generation of citations directly from the data. We believe this is important not only for consistency and accuracy, but also for efficiency. It may be better to generate citations "on the fly" than to store them in the database. The system was originally sketched in in [3] but we have added important extensions that provide location information of the citable units themselves and of nodes within the citable units.

In the current system, those nodes that correspond to citable units are tagged with a rule (or reference to a rule) that can be used to generate citations. For EAD we exploited the `<prefercite>` that is intended to contain a textual citation. In order to avoid any conflicts we specify an attribute (`<prefercite type="citationRule">`) that allows us to distinguish between an element used for common textual citation and the element containing a citation rule that will be interpreted by the citation system. In the IUPHAR XML serialization we use a `<citation>` tag with no attribute because there is no risk of ambiguity with other elements. Another possibility would be not to modify the XML hierarchy but to use XPATH that can be generated from the rules themselves to describe the citable units, but this introduces some complications and we leave it for later work.

The form of a rule suggested in [3] is $C \leftarrow P$ where $C$ provides the concrete syntax of a human or machine-readable citation and $P$ is a pattern that is expressed in the syntax of XPath augmented with decorated variables. The purpose of $P$ is not, as in XPath, to identify nodes, but to bind these variables that can then be used in $C$. The syntax of $C$ is not important here: there are literally hundreds of conventions for human-readable citations [6] and a good number of machine-readable forms. For illustration we shall use a simple name-value pair syntax: $\{a_1 = \$x_1, \ldots, a_n = \$x_n\}$ in which the variables $\$x_1, \ldots, \$x_n$ will be instantiated by the rule to produce, for example, {Author="Smith", Title="...", ...}.

The general form of a pattern is $P = /t_1[p_1^1 = \$x_1^1, \ldots, p_1^{k_1} = \$x_1^{k_1}]/\ldots/t_n[p_n^1 = \$x_n^1, \ldots, p_n^{k_n} = \$x_n^{k_n}]$ in which the $t_i$ are tag names and the $p_i^k$ are fully specified downward paths consisting of a sequence of tag names. The variables $\$x_1^1, \ldots, \$x_1^{k_n}, \ldots, \$x_n^1, \ldots, \$x_n^{k_n}$ are all distinct. Although this follows the syntax of XPath, its purpose is different. XPath identifies nodes in an XML tree. This pattern assumes that the node is given, and it binds values – XML fragments that are typically character strings – to these variables. Of course

[6] `http://citationstyles.org/styles` claims more than a thousand

we expect the given node to be specified by the XPath expression $/t_1/\ldots/t_n$, but this is where our first kind of variable plays a role. *Key* variables, denoted by $\$'$ are there to specify location information. If we remove all other variables from a pattern, to leave a pattern of the form $/t_1[p_1^1 = \$'x_1^1, p_1^2 = \$'x_1^2 \ldots]/t_2[\ldots]/\ldots$ we expect there to be a unique binding $v_i^j$ for each variable $'x_i^j$. That is, each of the paths $p_i^j$ is unique from its context node. More importantly, after substitution of these values, the XPath expression $/t_1[p_1^1 = v_1^1, p_1^2 = v_1^2 \ldots]/t_2[\ldots]/\ldots$ *uniquely* specifies the node at which the citation rule is attached. In other words, these bindings, together with the citation rule, provide location information[7].

It is assumed that the bindings of the key variables will be available and possibly expressed as an XPath expression in the citation $C$. We now describe to other useful bindings of variables. While for key variables we expect the paths $p_i^j$ to specify a node uniquely, in general such a path will specify a *set* of nodes. If we take document order, there is a list $V_i^j$ of XML values associated with each such path. The following decorations are constraints on $V$ that we expect to be useful in generating citations.

- $\$.x_i^j$: $|V_i^j| = 1$. The path $p_i^j$ is unique, and $.x_i^j$ is bound to the single member of $V_i^j$. One might, for example, insist on a unique title that is to be expressed in the citation. Note that there is no implication that a $\$.$ variable will serve as a key.
- $\$?x_i^j$: $|V_i^j| \leq 1$. Here, $?x_i^j$ is bound to the unique member of $V_i^j$ or to some designated null value if $V_i^j$ is empty.
- $\$ * x_i^j$: No constraints; $*x_i^j$ is bound to the list $V_i^j$. A possibly empty list of keywords, for example.
- $\$ + x_i^j$: $|V_i^j| \geq 1$. Again, $+x_i^j$ is bound to the non-empty list $V_i^j$. A non-empty list of authors for example.

With a little cunning these constraints can be checked in a single scan of the XML tree. The time taken to generate a citation will depend on the physical representation of the XML; in the worst case it can again be done with a linear scan through the whole document, but it can obviously be performed more efficiently if, for example, the document is represented as a main-memory tree. As a first example of the citation system at work, in EAD we considered each element representing a node in the archival hierarchy to be a citable unit. In Figure 3 we can see a set of rules that generates a machine-readable citation and an example of what it generates; the input of the system is the XML node representing the citable unit to be cited.

We can see that we exploit the hierarchical structure of the EAD file to build a citation and we use only the information already present in the document. When we have decided which archival element we need to cite, the system starts creating the XPaths from the rule corresponding to that element and then going up generating absolute XPaths from the rules of each ancestor element. The citation is created by executing the conjunction of the recursively generated XPaths. Finally, from the citation we can reconstruct the path from the document root to the cited element reversing the mechanism (i.e. $C \rightarrow P$). By means of this procedure we do not need any further metadata specifically created for the data citation. Furthermore, the system can be easily customized because the citation rules can be defined and tuned by the publishers who decide which information should be presented in the citation and how the citation should be defined.

Figure 4 is taken from IUPHAR and shows an example of how different rules are invoked for citable units at two levels of granularity. In contrast to EAD, in IUPHAR, we may want to cite nodes that are not citable units. The specification of a citation is exactly as before, with a pattern of the general form: $/t_1[p_1^1 = \$x_1^1, \ldots, p_1^{k_1} = \$x_1^{k_1}]/\ldots/t_n[p_n^1 = \$x_n^1, \ldots, p_n^{k_n} = \$x_n^{k_n}]$ which must include at least enough key ($\$'x$) variables to specify a key for that node, precisely as was required before. The system now searches up the path to the root to find the lowest citable unit and combines (with appropriate renaming of variables) the two patterns. Note that on the key variables, the two patterns must agree at any node that occurs at or above the citable unit. Now $C$ will contain

[7] In the presence of a key constraint [4] this check could be performed against the schema rather than the data

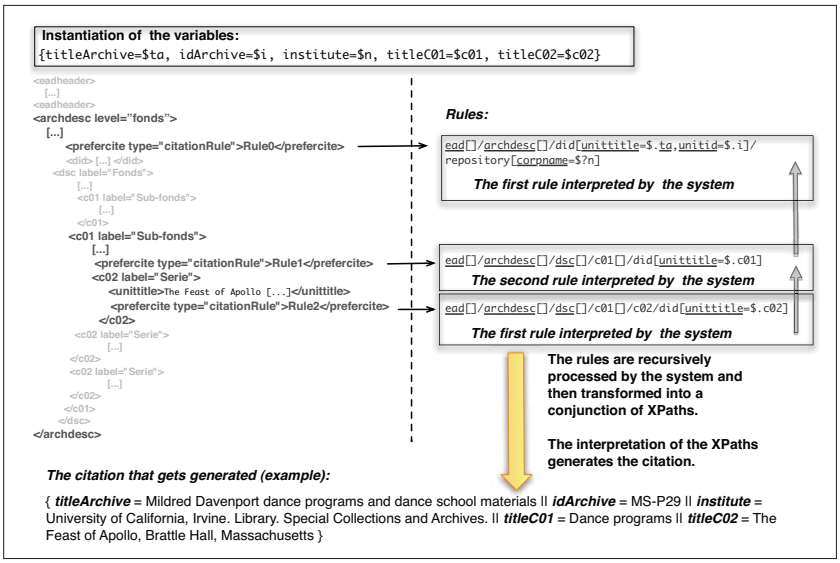



Figure 3: EAD: A set of rules that generates a machine-readable citation and an example of what it generates.

all the information for the citable unit in addition to some extra information about the part of that unit that is of interest. At this point we should observe that our simple name/value pair concrete syntax for the expression of citations is probably inadequate. A more sophisticated syntax may be required to delineate location information and what parts of a citation refer to the citable unit.

Note that if, as is the case in IUPHAR, the document is "fully keyed" [4]; that is, there is already a defined key for every node in the document, there is no overhead for adding location information for nodes that are not citable units. As an example suppose that we want to cite a specific datum within a ligand in an agonists table: we want to cite the fact that the action of the ligand named "CT (salmon)" (i.e. id=2097) is to be a "full agonist"; then, once the system has created the citation for the receptor (i.e. id="43") containing the ligand, it adds $E'$ = `/agonists[]/ligand[$.i]/action[]` that once interpreted generates the following relative XPath: `/agonists/ligand[id=2097]/action` allowing the system to cite the specific datum and then to reach the cited datum from the citation. As we have seen by means of this system we do not have to define a specific rule for each datum someone may want to cite, but only for a restricted set of citable units. Every part of a dataset is then citable without requiring any additional effort.

The presented system is compliant with versioning systems which take track of the changes of a dataset; in particular we can adopt the archiving system presented in [5] and implemented by Heiko Muller's XArch[8]. As we have stated before the EAD files are not very prone to change, instead this is the case of the IUPHAR database in which corrections are made. Our system can provide in the citation a version number, for instance $\{$*database=IUPHAR-DB, Version=15, Family=Glucagon receptor family*$\}$. The version number allows the system to select the correct snapshot of the database stored in the archival system and than to retrieve the cited element in the form in which it was cited. In this paper, as well as in [3], we chose to use a version number to record it at the database level (i.e. the coarsest citable unit). The temporal issues of data citation lead to some interesting questions (e.g. Why do we use a version number and not time? Why should the version number be recorded at the coarsest level?) that we shall further investigate providing an extension of the presented system comprising the temporal dimension.

[8] `http://www.dcc.ac.uk/resources/tools-and-applications/xarch`

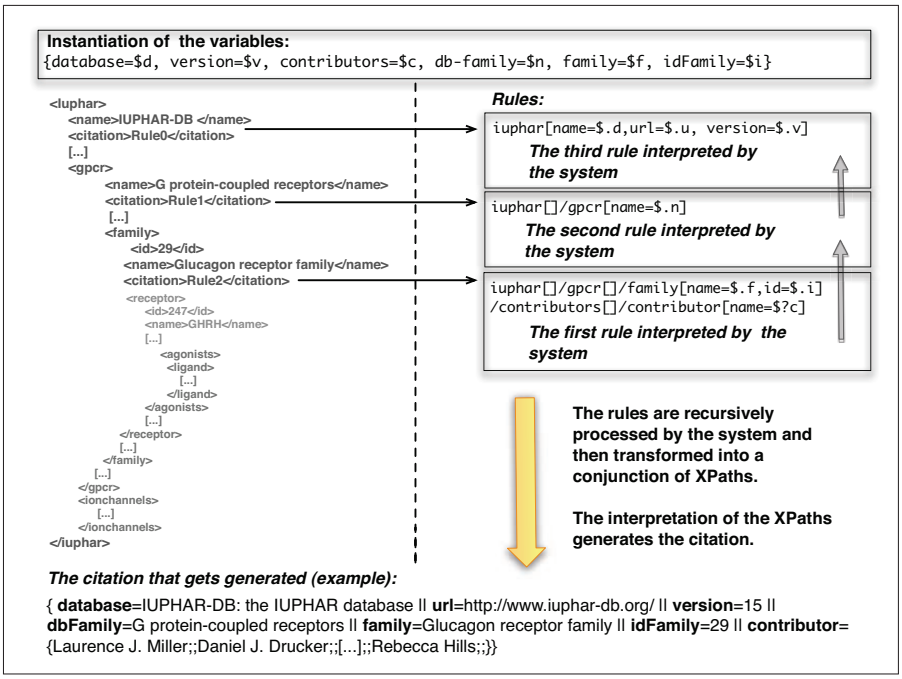



Figure 4: IUPHAR: A set of rules that generates a machine-readable citation and an example of what it generates.

## 5 Final Remarks

In this paper we outlined the main requirements a citation system must fulfill in order to cite structured and evolving datasets. We have argued that persistent object identifiers alone do not satisify the the conventional requirements of citations and have shown how a simple rule-based system can be used both to extract location information and other relevant data from a hierarchically structured data set.

An assumption we have made is that the data set is hierarchical and that this hierarchy is intimately connected with the structure of citations. This is certainly true for most scientific data sets. However there is a growing movement to express data by "linking": that is, through ontologies in which the underlying structure is a large graph and the notion of a citable unit or authorship is not immediately obvious. There are some proposals [7] to place a layer on top of this graph in order to deal with issues of provenance and trust. Whether this will lead to the hierarchical structures that are suitable for defining citations or whether ontologies will give rise to a form of scholarship in which citations are not needed remains to be seen.

## References

[1] M. Altman and G. King. A Proposed Standard for the Scholarly Citation of Quantitative Data. *D-Lib Magazine*, 13(3/4), March/April 2007.

[2] T. Arnold-Moore and R. Sacks-Davis. Databases of Legislation: the Problems of Consolidations. Technical report, Collaborative Information Technology Research Institute (CITRI), 1994.

[3] P. Buneman. How to cite curated databases and how to make them citable. In *Proc. of 18th Int. Conf.: on Scientific and Statistical Database Management, SSDBM 2006*, pages 195–203. IEEE Comp. Soc., 2006.

[4] P. Buneman, S. B. Davidson, W. Fan, C. S. Hara, and W. C. Tan. Keys for xml. *Computer Networks*, 39(5):473–487, 2002.

[5] P. Buneman, S. Khanna, K. Tajima, and W. C. Tan. Archiving Scientific Data. *ACM Trans. Database Syst.*, 29:2–42, 2004.

[6] S. Callaghan, F. Hewer, S. Pepler, P. Hardaker, and A. Gadian. Overlay Journals and Data Publishing in the Meteorological Sciences. *Ariadne*, 2009.

[7] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named Graphs, Provenance and Trust. In *Proc. of the 14th Int. Conf. on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[8] T. Green. We Need Publishing Standards for Datasets and Data Tables. OECD Publishing White Paper, Org. for Economic Co-operation and Development, 2010.

[9] J. Klump, R. Bertelmann, J. Brase, M. Diepenbroek, H. Grobe, H. Höck, M. Lautenschlager, U. Schindler, I. Sens, and J. Wächter. Data Publication in the Open Access Initiative. *Data Science Jour.*, 5:79–83, 2006.

[10] M. B. Line and A. Sandison. Progress in Documentation: 'Obsolescence' and Changes in the Use of Literature with Time. *J. of Docum.*, 30(3):283–350, 1974.

[11] D. V. Pitti. Encoded Archival Description. An Introduction and Overview. *D-Lib Magazine*, 5(11), 1999.

# Panda: A System for Provenance and Data*

Robert Ikeda
Stanford University
rmikeda@cs.stanford.edu

Jennifer Widom
Stanford University
widom@cs.stanford.edu

**Abstract**

*Panda (for Provenance and Data) is a new project whose goal is to develop a general-purpose system that unifies concepts from existing provenance systems and overcomes some limitations in them. Panda is designed for "data-oriented workflows," fully integrating data-based and process-based provenance. Panda's provenance model will support a full range from fine-grained to coarse-grained provenance. Panda will provide a set of built-in operators for exploiting provenance after it has been captured, and an ad-hoc query language over provenance together with data. The processing nodes in Panda's workflows can vary from well-understood relational transformations, to "semi-opaque" transformations with a few known properties, to fully-opaque "black boxes." A theme in Panda is to take advantage of transformation knowledge when present, but to degrade gracefully when less information is available. Panda yields interesting optimization problems, including data caching decisions and eager vs. lazy provenance capture. This paper is largely an overview of motivation and plans for the project, with some material on current progress and results.*

## 1 Introduction

In its most general form, *provenance* (also sometimes called *lineage*) captures where data came from, how it was derived, manipulated, and combined, and how it has been updated over time. Provenance can serve a number of important functions:

- **Explanation.** Users may be particularly interested in or wary of specific portions of a derived data set. Provenance supports "drilling down" to examine the sources and evolution of data elements of interest, enabling a deeper understanding of the data.
- **Verification.** Derived data may appear suspect—due to possible bugs in data processing and manipulation, because the data may be stale, or even due to maliciousness. Provenance enables auditing how data was produced, either for verifying its correctness, or for identifying the erroneous or outdated source data or processing nodes that are responsible for erroneous or outdated output data.
- **Recomputation.** Having found outdated or incorrect source data, or processing nodes that are buggy or have been modified, we may want to propagate corrections or changes to all "downstream" data that are affected. Provenance helps us recompute only those data elements that are affected by corrections or modifications.

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

*Work supported by the National Science Foundation under grants IIS-0414762 and IIS-0904497 and by the Boeing Corporation.

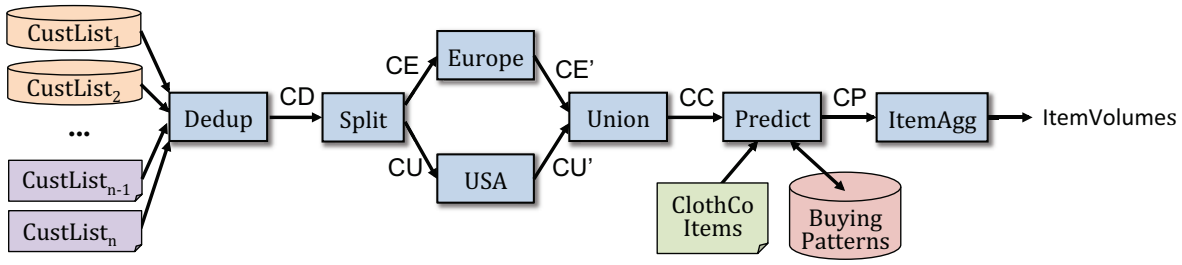



Figure 1: Analytics workflow example.

There has been a large body of very interesting work in lineage and provenance over the past two decades. Space limitations preclude detailed discussion of previous work here, but surveys are presented in, e.g., [2, 3, 5]. Despite all the past work, we believe there are still many limitations and open areas. Specifically:

1. Most work has been either: *data-based* (fine-grained), in which provenance of data elements is tracked based on well-defined, transparent properties of data models and query languages; or *process-based* (coarse-grained), in which provenance typically involves workflows and data at the schema level.
2. Often the primary focus is on *modeling* and *capturing* provenance: How is provenance information represented? How is it generated? There has been far less work on *querying* provenance: What can we do with provenance information once we've captured it?
3. Many projects have focused on specific functions or application domains, rather than developing a general provenance system that can be used for different purposes and across domains.

In the *Panda* (for "*Provenance and data*") project at Stanford, our goal is to fill these gaps. Specifically, in Panda we will:

1. Seamlessly merge data-based and process-based provenance, so that the two types of provenance can be combined (e.g., workflows that combine "opaque" processing nodes with well-understood relational queries and transformations). Thus, our model and system will offer users a full range from fine-grained to coarse-grained provenance.
2. Define a set of useful operators for taking advantage of provenance after it has been captured, as well as a general-purpose language for querying and analyzing provenance, and for combining provenance with relevant data.
3. Develop a general-purpose open-source system that is flexible and configurable enough to be used for a wide variety of applications. The system will support its own mechanisms for provenance capture, storage, operators, and queries, while also offering interfaces for coupling with outside data sources, processes, and systems.

## 2 Running Example

We use a detailed fictitious example to illustrate many of our goals for the Panda system. Consider *ClothCo*, a mail-order clothing company that is trying to decide which items to feature most prominently in its upcoming catalog. ClothCo runs an analytics workflow to predict which items will have the largest demand among its customers. The workflow is shown in Figure 1.

The initial input to the workflow is customer lists $\mathbf{CustList_1}$, $\mathbf{CustList_2}$, ..., $\mathbf{CustList_n}$, obtained from ClothCo's own databases and from partnerships with other companies. The lists contain names, addresses, and possibly additional attributes (e.g., gender, income), all of which are used for list deduplication and for buying-behavior predictions. The workflow involves the following steps:

- **Dedup**: Records from the customer lists are *deduplicated* so that customers represented in multiple lists are counted only once.
- **Split**: The deduplicated customer records (CD) are then partitioned into a European customer set (CE) and a USA customer set (CU). (If **Split** cannot determine that a customer is European, by default it assumes the customer is American.)
- **Europe / USA / Union**: Based on **Split**, records are routed either to **Europe** or **USA**, where the records' addresses are "canonicalized" into a standard form using existing software specialized for the region. The output lists CE' and CU' are unioned back to form canonicalized customer list CC.
- **Predict**: The next stage in the workflow predicts which ClothCo items customers are most likely to purchase. Specifically, for each customer in input list CC, and for each item ClothCo stocks, the **Predict** module consults a **Buying Patterns** database and attaches to the customer-item pair a likelihood that the customer will buy the item. **Predict**'s output CP is a set of customer-item-probability triples.
- **ItemAgg**: Finally, from the CP triples, **ItemAgg** aggregates the predicted demand for each of ClothCo's items. The output, ItemVolumes, contains a list of items and predicted sales volumes for each one.

Now suppose a ClothCo analyst runs the workflow and is surprised to find that the item in ItemVolumes predicted to have the highest demand is a cowboy hat, despite the fact that ClothCo's target customers rarely hail from the southern United States. Noticing this anomalous result, the analyst would like to find out why the predicted demand for cowboy hats is so high.

The Panda system will support the following interaction. Tracing the provenance of the cowboy-hat record in ItemVolumes back one step to CP yields a large data set, but with a simple provenance query the analyst discovers that the majority of the customers predicted to want the cowboy hat live in Paris, Texas. Something is clearly wrong: the population of Paris, Texas is only 25,000, and ClothCo doesn't cater to this demographic anyway. Further tracing the provenance of the relevant CP records, the analyst discovers that most of them were processed by **USA**, but they came originally from a French customer list. It turns out because the French list did not specify "France" explicitly in its addresses, **Split** mistakenly routed these customer records to **USA**, which added "Texas" during the canonicalization process. To fix the error, a simple new module is inserted that appends "France" to the addresses from the problematic list. Using provenance, when the modified workflow is rerun with the new module, only the item predictions potentially affected by the modifications are recomputed.

While this example captures a large number of the most important capabilities planned for the Panda system, there are a few missing. For example, we want to capture provenance from *human-generated* data edits and manipulations, along with computer-generated ones. Also, we expect that capturing and managing *evolving versions* of data, including long-term and frequent evolutions, will be a critical feature.

## 3 Processing Nodes and Provenance Capture

The workflow in our example is representative of the variety of processing node types that Panda will support. Specifically, processing nodes may vary from fully transparent (e.g., relational queries), to partially transparent (e.g., one-to-one transformations that apply an opaque function or filter on each element), to true "black boxes" (e.g., private code, calls to external services). Whenever possible, we want to capture provenance automatically. In addition, we plan to define an interface by which any type of processing node (including a human) can write out provenance information in a well-specified uniform fashion.

In our example, the **Union** and **ItemAgg** nodes perform standard relational operations. For relational nodes, there is a great deal of past work that can be applied for capturing and tracing provenance automatically and efficiently (see [3] for a survey). Consider **ItemAgg**, for example, which is a standard relational *group-by aggregation* operator. As defined by past work, the provenance of an element $e$ output by **ItemAgg** consists of all elements in CP with the same *group-by* attribute as $e$.

Now consider **Dedup**, an opaque processing node. We cannot rely on the automatic methods for relational queries mentioned above to capture and trace provenance for **Dedup**. Either **Dedup** must be instrumented to write out provenance information as it executes (presumably in the form of mappings between deduplicated records and their original customer records), or it must provide some sort of procedure for provenance-tracing, or in the worst case we cannot determine fine-grained provenance at all.

One of the most important first steps in Panda is to formalize its provenance model. Possible models range from a simple bipartite graph structure connecting input and output data elements, to the much higher-level and more "semantic" *Open Provenance Model* [17]. We decided to begin with an implicit bipartite graph model induced by *provenance predicates* (Section 7). However, the model will need to be enhanced as the project develops, in keeping with our goals of combining data-based and process-based provenance, and capturing the range from fine-grained to coarse-grained provenance (recall Section 1). Based on the provenance model, Panda will specify a uniform interface by which all types of processing nodes can create and manipulate provenance, both manually and automatically.

Once provenance has been captured, what are we going to do with it? Our overall goal is to support the features mentioned in Section 1: explanation, verification, and recomputation. To do so, Panda will offer at least two methods of using provenance:

- A set of built-in *operations* that can be used on their own or as building blocks for higher-level functionality (Section 4)
- A full-featured *query language* that can be used to pose ad-hoc queries and analyses over provenance information, and over provenance information combined with relevant data (Section 5)

# 4 Provenance Operations

For basic built-in operations, Panda will support at least:

- **Backward tracing.** Given a derived data element $e$, where did $e$ come from? That is, what data elements and/or processing contributed to $e$? In our running example, we used backward tracing to go from the output cowboy-hat record $c$ to the record-set $\mathcal{R}$ in input list CP from which $c$ was derived. Then we used further backward tracing to determine that most records producing $\mathcal{R}$ came through the **USA** processing node and originated from a specific French customer list.
- **Forward tracing.** Given an input or derived data element $e$, where did $e$ subsequently go? That is, what processing nodes did $e$ later pass through and what data elements were produced by it? In our running example, we can use forward tracing to determine all of the item predictions that were affected by French customers erroneously passing through the **USA** processing nodes.

From these two basic operations, we can layer on additional functionality, for example:

- **Forward propagation.** If an input or derived data element $e$ changes, propagate the change to everything it affects. Clearly this function will rely on forward tracing. In our running example, once we correct the problem with our French customer list, we can use forward propagation to recalculate only those item predictions affected by the correction.
- **Refresh.** Given a derived data element $e$, check if $e$ is still valid. If it is not, refresh it to its new valid value. Clearly this function will rely on both backward tracing and forward propagation. In our running example, suppose we correct the error with the French addresses, but ironically a celebrity then begins wearing cowboy hats. Suddenly cowboy hat sales soar, with corresponding modifications in our **Buying Patterns** database. We decide to once again investigate the cowboy hat prediction in ItemVolumes, this time asking for it to be refreshed to ensure we get the latest predicted demand.

# 5 Provenance Queries

The operations in the previous section are not specific to a particular application, but they still encode specific provenance-based functionality. We believe that in addition to a set of common operations, it will be very useful for a general-purpose provenance system to support a declarative ad-hoc query language, similar to what is provided by database management systems.

As we develop our query language, we have a few guiding principles. One is that we want our query language to operate over provenance and data seamlessly, so that they can be combined in useful ways. Second, we want our query language to be compact and intuitive for basic queries, with specific features for additional expressiveness, similar in spirit to SQL. Finally, the language must be amenable to finding efficient query execution plans, again in a database-system style.

Our running example relied on a query that aggregated the data elements in CP contributing to the cowboy-hat output element in ItemVolumes. Here are a few other queries that demonstrate the type of functionality we plan to support in a query language:

- Determine which customer list contributes the most to the top 100 predicted items.
- Considering only customers from a specific list, which items have significantly higher demand among those customers than among the general population?
- Which customers have more duplication in our original customer lists—those that are eventually processed by **USA**, or those that are processed by **Europe**?

# 6 System and Optimization Issues

Building a full-featured, general-purpose system for managing provenance and data together entails a significant effort just for the basic functions: provenance capture and storage, built-in operations like those in Section 4, and general-purpose query processing. (Progress on our initial prototype is described later in Section 7.2.) In addition, we see a number of specific opportunities and challenges, both theoretical and systems-related.

### Query-Driven Capture

The most general approach is to capture provenance in support of any possible operation or query that may subsequently be performed using it. However, if there is a known restricted set of operations and queries to be performed, we may be able to streamline what is captured.

### Eager vs. Lazy

Even when supporting arbitrary operations and queries, there may be a choice between *eager* and *lazy* provenance capture. It's a typical space-time tradeoff: eager capture occupies space but speeds up operations and queries. (Note that capturing all possible provenance at a fine-grained level could entail enormous amounts of space.) Also, similar to materialized views, eager capture may incur an update cost if we wish to keep provenance up-to-date. The choice between eager and lazy might be made on a per-processing-node basis. For example, eager capture makes sense for **Dedup**, since the overhead of recording the source customer records contributing to each deduplicated record may be fairly low, while recomputing that information may be very expensive (perhaps requiring deduplicating the entire data set again). On the other hand, for **ItemAgg**, if we have access to intermediate data set CP, then computing the provenance of an output element in **ItemAgg** is a simple selection query over CP.

#### Intermediate Results

Along similar lines, in a workflow such as our ClothCo example, we may or may not wish to store all of the intermediate data sets. Storing these data sets may be very helpful for provenance operations (such as finding the provenance of an **ItemAgg** output element by querying CP, as above) but the storage overhead may be high, and intermediate results may not always be necessary. For example, if we only wish to consider provenance from final output elements back to initial input data, and we take a fully eager approach, then no intermediate data sets are necessary. Overall, we envision a suite of interesting optimization problems involving decisions about intermediate data sets and eager vs. lazy computation.

#### Fine-Grained vs. Coarse-Grained

Another trade-off, perhaps intertwined with the previous two, is between fine-grained and coarse-grained provenance. Even if it is prohibitive to compute provenance eagerly at the data-element level, it may be helpful to record some provenance at the data-set level: some operations and queries may only need coarse-grained provenance, or perhaps coarse-grained provenance can be used to support later computation of fine-grained provenance. Suppose, for example, for each input list we record only the percentage of its customers that are eventually sent to processing node **Europe**, versus those sent to processing node **USA**. Queries may ask for this information directly, or in some cases (e.g., lists that are 100% **Europe** or 100% **USA**), this schema-level information may be used to optimize provenance queries involving individual records.

#### Approximation

Another avenue for dealing with prohibitively large fine-grained provenance may be to support *approximate* provenance capture, operations, and/or query results.

## 7 Initial Progress

We briefly describe some of our initial progress. We decided to drive our work by considering one of the specific operations discussed in Section 4, *refresh*. Targeting an "end-to-end" efficient solution to the refresh problem forced us to: become more concrete about the *data-oriented workflows* Panda is targeting; to develop formal underpinnings and a number of algorithms; and to code up a fledgling prototype system; all of which are now serving us well as a basis for the entire Panda project.

### 7.1 The Refresh Problem

Consider any workflow that is an acyclic graph of processing nodes, as in our running example. Suppose the input data sets have been modified since the workflow was run, but the workflow has not been rerun on the modified input. However, suppose we would like to *selectively refresh* one or more elements in the output data, i.e., compute the latest values of particular output elements based on the modified input data. With selective refresh, we can avoid the expense of recomputing an entire output data set frequently, when all we need is a few up-to-date output values after the input has undergone some changes.

Our initial approach to performing selective refresh assumes that we capture input-output data provenance at each processing node when the workflow is run initially, although lazy processing and optimizations like those discussed in Section 6 are certainly possible. In our initial approach, provenance takes the form of *provenance predicates*: Each output data element $o$ has an associated predicate, which is applied to the input sets to compute $o$'s provenance. For example, each element $o$ in ItemVolumes has a provenance predicate that selects all customer-item-probability triples from CP with the same item as $o$. Provenance predicates are flexible and

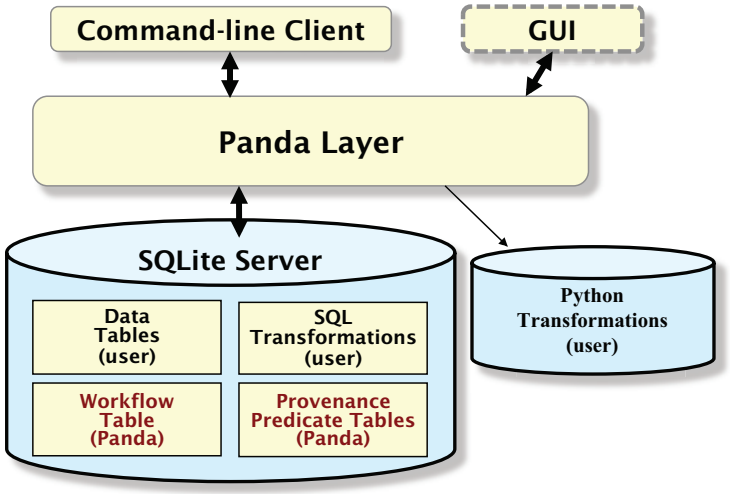



Figure 2: Architecture of the prototype Panda system.

general: Provenance in the form of pointers to specific data elements can be specified as predicates selecting on identifiers or keys, and provenance for basic relational operators [3] is easily expressed as predicates. Note that provenance predicates, when applied for all output elements, yield a bipartite graph structure as discussed in Section 3.

Frequently, provenance predicates take the same basic form for each data element in an output set, and we do plan to exploit that fact. For example, recently we added *attribute mappings* as a schema-level provenance specification mechanism. The provenance representation is very compact, while the semantics still logically yields a provenance predicate for each output element. For example, the provenance for ItemVolumes can be specified using an attribute mapping between CP.item and ItemVolumes.item. The element-level provenance predicates implied by this mapping are the same predicates discussed in the previous paragraph.

To perform refresh, we first apply provenance predicates recursively to determine which input elements are responsible for the output elements to be refreshed, i.e., we perform *backward tracing* as discussed in Section 4. Then we rerun the workflow only on that portion of the data needed for refresh. i.e., *forward propagation* from Section 4. Laying the formal foundations, developing the algorithms, and implementing refresh in a prototype system was a substantial challenge. One of our important results was to identify specific properties of processing nodes, provenance, and workflows for which refresh can be very efficient, while it is inherently less efficient when the properties do not hold. For details, please see our technical report [4].

### 7.2 Panda System Prototype

We began by building up the system infrastructure and features needed to experiment with all aspects of *refresh* as described in the previous section. In doing so, we produced an excellent starting point for the more ambitious and general-purpose Panda system.

The high-level architecture of the Panda system is shown in Figure 2. For now, all data must be representable as relational tables, and the main backend is a **SQLite** server. The server stores all data sets, relational (SQL) transformations, provenance, and workflow information. Panda also supports opaque processing nodes programmed in Python; they are stored separately in files.

Panda supports arbitrary acyclic graphs of processing nodes. Currently, users interact with Panda through a simple command-line interface; a GUI is underway. There are three general types of user commands: (1) Creating or modifying input tables; (2) Creating transformations that generate newly-defined tables from existing ones, to build up workflows; (3) Backward-tracing and refresh commands. When workflows are created and run,

Panda stores everything needed to support selective refresh: provenance and intermediate data sets for backward tracing, and transformations for forward propagation. Panda currently supports provenance predicates as described in Section 7.1; provenance based on schema-level attribute mappings (Section 7.1) is also underway.

Panda supports transformations specified as SQL queries, including queries involving multiple input tables. Provenance predicates are created automatically for SQL transformations, following known definitions and techniques [3]. For Python processing nodes, Panda currently supports one-one and one-many transformations only, so that provenance predicates can be generated automatically. Specifically, the "opaque" Python code associated with a processing node is run separately on each data element in the input set (multi-input Python transformations currently are not supported), producing zero or more elements for the output set. Input data elements are required to have keys, so the provenance predicate for an output element can simply select the key of the corresponding input element. Support for more general Python processing nodes is in progress.

## Conclusions, Website, and Acknowledgments

The Panda project is in its early stages; much of the work is ahead of us. We have a fairly clear roadmap in terms of motivation, goals, and specific technical challenges, as laid out in Sections 1–6, and we have a solid start with the work reported in Section 7. For more information, and to follow progress in the Panda project, please visit our website: `http://i.stanford.edu/panda`.

We are grateful to Parag Agrawal, Charlie Fang, Diana MacLean, Abhijeet Mohapatra, Raghotham Murthy, Aditya Parameswaran, Hyunjung Park, Alkis Polyzotis, Semih Salihoglu, and Satoshi Torikai for many useful Panda meeting discussions that have been instrumental in forming the direction for our system. We thank Semih Salihoglu for his contributions to the refresh work and the Panda system, and we thank Charlie Fang and Satoshi Torikai for their contributions to the Panda system. We thank Philip Guo for his helpful comments on this paper.

## References

[1] The Open Provenance Model (v1.01). July 2008. `http://eprints.ecs.soton.ac.uk/16148/`.

[2] BOSE, R., AND FREW, J. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv. 37*, 1 (2005), 1–28.

[3] CHENEY, J., CHITICARIU, L., AND TAN, W.-C. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases 1*, 4 (2009), 379–474.

[4] IKEDA, R., SALIHOGLU, S., AND WIDOM, J. Provenance-based refresh in data-oriented workflows. Technical report, Stanford University, 2010.

[5] SIMMHAN, Y. L., PLALE, B., AND GANNON, D. A survey of data provenance in e-science. *SIGMOD Rec. 34*, 3 (2005), 31–36.

# Provenance for Scientific Workflows Towards Reproducible Research

Roger Barga[1], Yogesh Simmhan[1], Eran Chinthaka Withana[2], Satya Sahoo[3], Jared Jackson[1], Nelson Araujo[1]
[1]Microsoft Research, Redmond WA 98052
[2]Indiana University, Bloomington IN 47405
[3]Wright State University, Dayton OH 45435

## 1 Introduction

eScience has established itself as a key pillar in scientific discovery, continuing the evolution of the scientific discovery process from theoretical to empirical to computational science [13]. Extensive deployment of instruments and sensors that observe the physical and biological world are bringing in large and diverse data to the reach of scientists. Often, that data is more frequently shared due to the cost of the instrumentation or because of the desire to address larger scale and/or cross-discipline science such as climate change. There is a tangible push towards building large, global-scale instruments [2][3] and wide deployment of sensors [1] with the data they generate being shared by a large collaboration which has access to the data generated by these instruments. Indeed, funding agencies and publishers are starting to insist that scientists share both results and raw datasets, along with the provenance for how the result was produced from the raw dataset(s), to foster open science [4].

Scientific workflows have emerged as the de facto model for researchers to process, transform and analyze scientific data. These workflows may run on the users desktop or in the Cloud and the workflow framework is geared towards easy composition of scientific experiments, allocation and scheduling of resources, orchestration and monitoring of execution, and collecting provenance [20]. The goal of the Trident Scientific Workflow System is to provide a specialized programming environment to simplify the programming effort required by scientists to orchestrate a computational science experiment.

### 1.1 Trident Scientific Workflow System

In designing Trident [9][5], our goal was to leverage the existing functionality of a commercial workflow management system to the extent possible and focus development efforts only on functionality required to support scientific workflows. The result is a much smaller code base to maintain, improving sustainability, and a thorough understanding of requirements unique to scientific workflows. Trident is implemented on top of Windows Workflow Foundation [6], a workflow enactment engine included in the Windows operating system. Windows Workflow Foundation is highly extensible. All activities in Windows WF derive from a common .NET base class and an extensible development model enables the creation of domain specific activities that can then be used to compose workflows useful and understandable to domain scientists. In addition users can add new infrastructure services to the workflow runtime, such as automatic provenance capture.

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

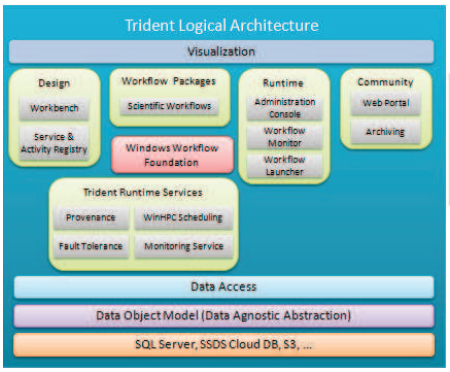



Figure 1: Diagram of the Trident logical architecture

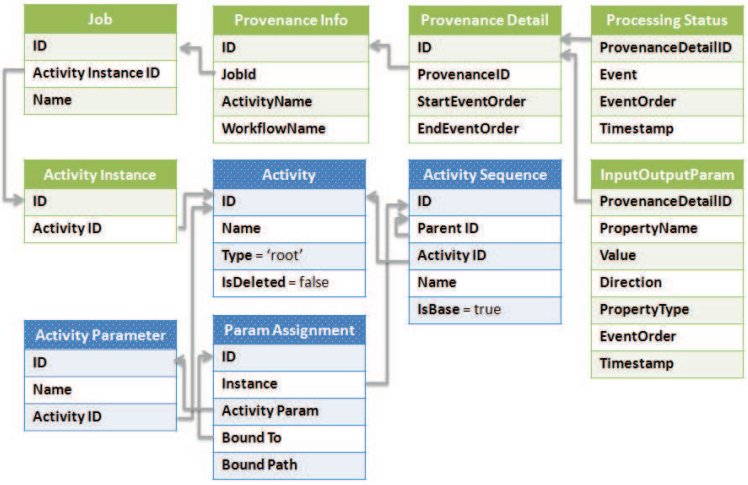



Figure 2: Provenance Data Model. Entities in dotted blue describe the workflow schema that is populated during composition. Entities in solid green describe the provenance schema that is populated at runtime.

The key elements of the Trident architecture, illustrated in Figure 1, include a composer that enables users to visually author a workflow using a library of existing activities and complete workflows. The registry serves as a metadata catalog of known datasets, services, workflows, activities, and computational resources, and it maintains state for workflows that are currently active. Administrative tools allow users to register and manage compute resources, track workflows currently running, along with a workflow launcher that supports scheduling of workflows. Community tools include a web service to launch workflows from any web browser, access to a repository of workflow results, and a facility to publish and share workflows with other scientists. At the lowest level of Trident is a data access layer that abstracts the actual storage service used from the workflows, so a workflow can read and/or write data objects that are transparently mapped to the target data source. We currently support a default XML store and SQL Server for local storage, along with Amazon S3 and SQL Azure for cloud storage.

The rest of the paper is organized as follows: Section 2 describes the Trident provenance data model, along with the capture, storage and querying of provenance, while Section 2.4 describes the workflow evolution framework in Trident that tracks the evolution of workflow definition and the results produced by individual workflow version, related work is presented in Section 4, and we summarize our paper in Section 5.

## 2 Provenance in Trident

The Trident Workbench provides an integrated way to collect, store, query, and view provenance for scientific workflows [18]. Windows WF generates low level events during every step of workflow execution, from the point a workflow is loaded for execution to workflow termination. Trident intercepts and routes these event through a publish-subscribe API; C# libraries we provide allow user defined provenance events to be published. Trident enhances basic Windows WF tracking by adding automated logging of workflow input and output parameters, execution events and data flows, including external applications launched from the workflow and system calls. In addition, Trident tracks workflow evolution provenance [14], to record changes to the individual workflows over time and correlates versioned workflows with the results they generate [8]. The provenance data collected by Trident is compatible with the emerging Open Provenance Model standard [17] and different

data stores are supported for storing provenance. In the following sections, we discuss the interoperable data model used to represent provenance in Trident, means for distributed collection of provenance from the various workbench components, and its storage and dissemination.

## 2.1 Provenance Data Model

Provenance information is available as a combination of static, composition information about the workflow - *the workflow schema*, along with dynamic, runtime information about the actual execution of an instance of a workflow - the *provenance schema*. The workflow composition information provides structural knowledge of the workflow, its activities, and their data and parameter type signatures, and is static for a composed workflow version. These are described using the *Activity, Activity Sequence, Activity Parameter* and *Parameter Assignment* entities in the data model shown in Figure 2. The composed workflow is an abstract workflow that can be instantiated with initial parameter and data values, and executed by the workflow engine. Information about the actual workflow execution forms the runtime provenance, and includes the job that represents a local or remote submission of the workflow instance to the workflow engine, the workflow and activity execution times, the parameters to/from each activity instance, and the status of their execution. The *Job, Activity Instance, Provenance Info, Provenance Detail, Processing Status* and *InputOutputParam* entities in Figure 2 record these. Using this two level model of abstract workflow and workflow instance details intuitively corresponds to composition and execution phases of a workflow. It also allows common workflow features that do not change across runs to be stored once while dynamic information unique to each workflow run is accumulated. Having explicit relations between the abstract and instance metadata allows rich queries that combine both workflow structural features and runtime knowledge.

This native data model used in Trident is also compatible with the Open Provenance Model (OPM) specification [17] that evolved at the same time as Trident. OPM allows interoperability with other provenance and workflow systems to allow provenance tracking of data that are reused across workflow systems, as is common in collaborative research projects. Both OPM and Trident use similar entities to represent workflows and activities (OPM Processes), data products and parameters (OPM Artifacts), and relationships like Used and Produced for data dependencies. The OPM model is a subset of the native Trident model and we have demonstrated the ability to export Trident provenance to OPM and import it from other systems to perform interoperable provenance queries, as part of the third Provenance Challenge workshop [19].

## 2.2 Provenance Collection

The primary source for the abstract workflow details is the workflow composer. Workflow activities imported into the workbench are examined and their signatures extracted into the Activity and Activity Parameter entities. When users compose new workflows or import existing ones and save them, the Activity Sequence and Parameter Assignment entities store the data and control flows present in the workflow.

Collecting runtime provenance information about a workflow instance is more challenging given the distributed nature of workflow execution. The different sources of dynamic information include the *execution service* that submits and triggers workflow job execution, the *Windows WF engine* that orchestrates the workflow instance by invoking activities and the *activities* themselves that may be built-in or user defined, illustrated in Figure 3. The execution service tracks the submission of each workflow instance by the workbench or management studio interface, its status (scheduled, executed, or completed) on a local or remote machine. It is the source for *Job* and *Activity Instance* entities of the data model. The Windows WF engine natively generates tracking events of the workflow execution, such as when the workflow starts and finishes, an activity initializes, executes, and completes successfully or fails, or an exception is raised. Trident event handlers to listen for these built-in events and populate the *Provenance Info, Provenance Detail* and *Processing Status* entities that track the workflow's control flow. Lastly, we use instrumentation in the Trident base activity class to generate custom

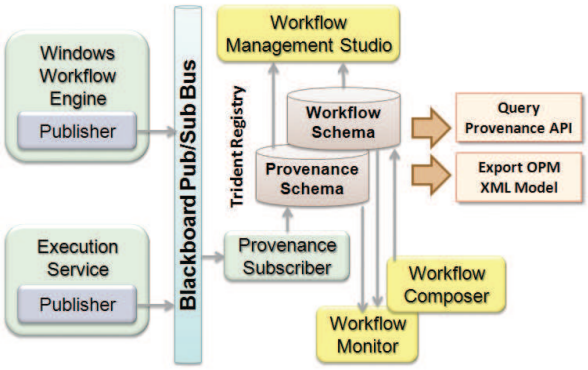



Figure 3: Provenance Architecture

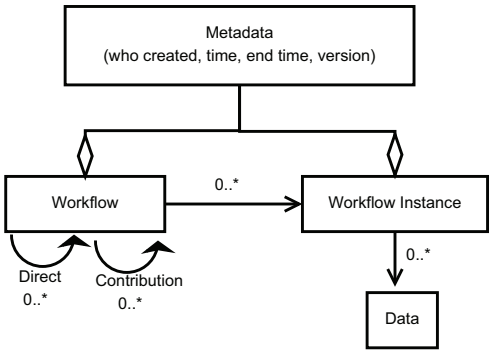



Figure 4: Versioning Data Model within EVF

user events that capture the actual input and output parameter values passed to/ from the activities that provide data flow knowledge used to fill the *InputOutputParam* entity.

Given that these information sources may each be running on a different machine, and remote from where the provenance is stored, we have developed the BlackBoard [21] publish-subscribe, asynchronous messaging framework to distribute provenance events from the sources to the provenance storage. The messages form incremental pieces of the complete provenance information. They are generated by publisher libraries incorporated into the workflow engine and execution Service, and are listened to by a Provenance Service subscriber. Message payloads consist of a list of name-value attributes that identify the workflow instance and activity that the message describes, the timestamp of the message, and details such as the current status of the workflow or activity, input or output parameter values, and exceptions that may have occurred. The provenance service listens to the events and records them in the provenance store in the Trident Registry.

Besides distributed provenance collection, the use of the BlackBoard pub-sub model has two other advantages: (1) it separates the responsibility of message creation by the source from its fault resilient delivery by the BlackBoard broker to the subscriber, and (2) it allows the same set of messages to be delivered to different subscribers with different goals, such as real-time monitoring in a GUI, filtering them to detect workflow failures or anomalies [3], or simply recording them in the provenance store. Both of these features are essential for reliable scaling of workflow instances across distributed resources [21].

## 2.3 Provenance Storage

The Trident Registry is the central information repository and stores the provenance and workflow metadata. It uses the data access layer to abstract the data model from the storage layer, allowing the use of local and cloud backends. The Registry data model can be easily configured through an XML schema file and supports entities and relationships with different cardinalities.

Trident's complete data model, including the provenance and workflow entities, is created and available in the Registry schema with auto-generated .NET objects for programmatic access all the entities. The provenance service and other Registry users initially create a .NET handle to the active workbench Registry. Inserting a tuple for a particular entity is as simple as instantiating a .NET object of that type and setting its properties. Similarly, relationships between different entity tuples are created from corresponding .NET object references. Depending on the Registry storage backend, storage adapter implementations map the .NET objects to the storage layer, for example, SQL Tables and tuples or Amazon S3 blob objects. This approach makes it easy for .NET services and workbench to easily and intuitively store data in the Registry without programming directly to the storage implementation, and allows backend storage to be switched to suit the needs of the scientific domain.

The Registry also supports queries over the data model using the .NET Language Independent Query (LINQ) mechanism [10]. This provides a collection abstraction for accessing stored items in the data model and allows filter, project, and aggregate operators over the collection. Specifying queries becomes a natural extension of the .NET language and results are returned as collections of .NET objects. LINQ also supports deferred, just in time execution for efficiency. The Registry's backend storage adapters provide the implementation for mapping the LINQ to the access/query model supported by the storage layer, such as, SQL or REST.

### 2.4 Provenance Query and Dissemination

Provenance is typically used in three ways in Trident: for realtime monitoring of the workflows, for analysis and mining post workflow execution and for exporting to external provenance systems. The workflow monitor GUI allows users to launch and visualize the workflow execution from a user's desktop. Upon workflow launch, the workflow monitor subscribes to provenance messages from the BlackBoard broker for that workflow instance. The light-weight provenance messages published by the workflow engine and execution service allow the workflow monitor to show the current status of each workflow activity as they execute. The monitor also displays resource performance information, such as CPU and memory usage, of the machine running the workflow and activities that are also published as separate events. Additionally, the workflow monitor can pull more detailed information about the workflow execution by querying the provenance service using concise LINQ queries over the Registry data model. This is also possible through the workflow management studio where a user can list of all past and currently executing workflows and retrieve details of the workflow, its execution, the data produced and consumed, and the faults that happened, and step through each activity execution in the workflow to visually replay the execution. This also allows the same workflow to be re-executed for validation. These features provide a powerful tool for scientists to maintain a comprehensive record of their in silico experiments that is useful for verification and reproducibility.

Some of the queries supported by the provenance and workflow data model include:

- *Display the **status statistics** of all **workflows** run **today***: This groups all workflows run today by their status property, and returns and prints the count for workflows for each status.

```
var jobStats =
  // Todays' WFs
  from j in jobs where j.Update.ToShortDate() == DateTime.Now.ToShortDate()
  // Group by status
  group j by j.Status into g
  // WF count for each status
  select new {status = g.Key, count = g.Count()};
// Print stats
foreach(var stats in jobStats) Console.WriteLine(stats.ToString());
```

- *Find the authors who **composed workflows** which have the output of the **SecretTask** activity **passed as input** to the **OnlinePublish***: This queries the data flow of the abstract workflow for those workflows that can potentially publish the output for a proprietary 'SecretTask' activity through an OnlinePublish activity, and identifies the authors of those workflows.

```
var badAuthors = from aseq in
  // Activity OnlinePublish's input is connected to output from activity
  // SecretTask
  (from pa in paramAssigns
   where pa.ActivityParameter.Activity.ActivityClass == "OnlinePublish" &&
         pa.BoundTo.ActivityParameter.Activity.ActivityClass == "SecretTask"
   select pa.ActivityParameter.Activity.ActivitySequences).SelectMany(sq => sq.ToList())
  where aseq.Parent.IsBase select aseq.Parent.Activity.Author; // Authors of parent WF
```

- *Find all **abstract workflows** that have been **executed** with an **activity** that uses the **Needleman-Wunsch** as the genome sequence **aligner** parameter*: This provenance data flow LINQ query selects InputOutput entities

with the 'aligner' activity parameter set to 'Needleman-Wunsch' and traces them back to jobs that ran the workflow instances containing that activity, and returns the abstract workflows for those jobs.

```
var needlemanWorkflows =
  // 1. Activity is a workflow, and ...
  from a in activities where a.IsWorkflow &&
    // 2.A. Job IDs for WF
    (from j in (from ai in a.Instances select ai.Jobs) select j.ID)
  .Intersect
    (from pi in (from io in inOuts where io.Name == "aligner" &&
                                         io.Value == "Needleman-Wunsch" select io.Provenance)
        // 2.B. Job IDs with Needleman-Wunsch activity parameter value
        select pi.JobId
    // Intersection of job IDs from (2.A) and (2.B) has items
    ).Count() > 0   select a;
```

Lastly, provenance for a workflow instance execution stored in the Registry can be published as an XML document following the OPM specification using a prototype tool. This allows workflow provenance to be interoperably shared with the external community for collaboration, for peer review, or as part of a publication that uses that experiment, usable by even users who do not employ Trident as their workflow workbench.

## 3 Versioning of Workflow Evolution

As researchers use a workflow management system to carry out their computations the workflows evolve as the research evolves and this workflow evolution can be a valuable tool for tracking both the evolution of the research and results created by a specific workflow instance across time. Scientists can trace their research and associated results through time or go back in time to a previous stage and fork a new branch of exploration. And since workflows encapsulate a vast amount of knowledge associated with experiments, tracking the evolution of workflow can help to aggregate this knowledge for later analysis. To support this, Trident provides a workflow evolution framework (EVF) to enables efficient management of knowledge associated with workflow evolution over time. The benefits of the Trident workflow evolution framework include the following:

- *Research evolution*: When a scientist associates their research with a collection of workflow, tracking the evolution of these workflows becomes an approximation to the initial problem of tracking the evolution of their research process. Along with the evolution of a workflow, all the components within it will also evolve from the selection of web services and databases to the implementation of individual activities. Scientists can later examine the result of a workflow execution and reason about how the research evolved to the current state to produce that particular output.
- *Result Comparison*: A given research exploration may evolve in more than one direction. It is important to understand the changes these choices had on the outcome of the research by comparing the difference between the outputs of two or more versions of the same workflow.
- *Attribution*: When a workflow is executed, attribution information such as who performed the experiment, the author of the workflow, the owner of the data products, etc., can be collected at runtime and associated with the final result. This attribution information can be used later to identify issues with data or code quality and to give proper credit to contributors. Also, while carrying out an experiment it is becoming more common to reuse subset graphs within a workflow scientists can utilize not only the algorithms and implementations developed by others, but also the data products generated including optimally derived model parameter configurations. In research, it is not only the technical aspects that matter; sharing and attribution of research can and should be an integral part of research. The Trident workflow management system can access and download subset graphs from sources like myexperiment.org [12] to reduce development costs and track the author with our evolution framework for proper accreditation to the contributors.

### 3.1 Versioning Model

In order for workflow based research to be reproducible, a versioning strategy needs to consider the workflow, along with the associated data products, parameters, configurations and executable, and bind this information together. The Trident Workflow Evolution framework can support reproducibility by persisting all information about previously executed experiments. If the underlying data management service supports versioned data products, then a scientist using Trident can re-run previously executed experiments.

This versioning model, illustrated in Figure 4, is built on two orthogonal dimensions of workflow evolution, namely direct evolution and contributions. Direct evolution occurs when a user performs one of the following:

- Changes the flow and arrangements of the components within the system;
- Changes the components within the workflow;
- Changes inputs and/or output parameters or configuration parameters to different components within the workflow.

Direct evolution will primarily come from a researchers direct manipulation (editing) of the workflow that is being tracked. On the other hand contributions will track components that are reused from previous system.

One of the unique features of the Trident EVF is that it tracks both direct evolution and contributions to research. Together this contributes towards the existing eco-systems to acknowledge each other's contributions to the existing research and also encourages scientists to share and use existing work. Versioning of workflows and related artifacts is done at three separate stages of execution.

- User explicitly saves the workflow;
- User closes the workflow editor;
- Executing a workflow in the editor: since workflow instances should always be associated with a workflow, EVF requires all the workflows to be saved and versioned before executing them.

This level of granularity does not capture minor edits to a workflow, but in applying EVF in actual use cases we have established a level of sufficiency for this versioning for later retrieval and workflow evolution. Figure 4 also presents the data model used for versioning of objects and the relationships between them. This model is designed such that all the artifacts related to an experiment can be captured and versioned.

### 3.2 Architectural Features

There are several architectural features that enable the workflow evolution tracking, which are presented below:
**Unique Association of Research Artifacts to Workflows** As a workflow is executed, the Trident EVF associates the relevant data, parameters, and configuration information, as well as meta-data identifying who performed the experiment (user id), when and where the output was saved in the system, etc. In addition Trident records the lineage information of the workflow. This information is of value in evaluating the end result and can be used to reproduce the research at a later time. Trident records this information in the provenance log using the information model outlined in Figure 4 and associates both provenance log and output result in the Trident Registry.
**Automatic Versioning** The Trident EVF automatically versions workflows as users edit them. This enables a researcher to later retrieve a previous workflow for viewing or to create new branches from previous points in the workflow evolution. Versioning of the workflow templates inside EVF is comparable to a typical version control system, but EVF also has the ability to work with other versioning systems to support different versions of data products. The Trident EVF provides clearly defined extension points to add new versioning systems. Once a data provider, capable of versioning data products, is registered with the system, EVF will save enough information to retrieve a given version of a data product. When EVF associates a data product with a workflow execution it will include this versioning information so that the correct version of the data product can be retrieved later, in case the scientist is interested in reproducing the research. Also if an extension is registered to handle versioning, EVF will use that extension to automatically retrieve the data and to execute the workflow within Trident. During the

workflow authoring process, the user may make multiple changes to a workflow and possibly save intermediate steps. But in the end only execute the final version of the workflow. Should the scientist opt to delete the previous versions, EVF gives the control to the user to select the versions to persist inside the registry or to remove from the system. This will not only reduce the clutter in the scientist's workspace, but also optimizes the workflow lineage information persistence. However, a user is not allowed to delete a workflow version that is either associated with a result or contributed to a workflow that produced a result.

**Navigation through Time** Navigating a workflow evolution through time can provide a unique view on the evolution of the research and associated output results. A user may see a change or improvement in the results of tan experiments over time, observe the effects of the different data sets being used, or identify the contributor of a new subset graph (workflow). Trident captures sufficient information to allow a user to select a workflow and navigate previous versions of that workflow through time, visualizing the evolution of both the workflow and associated results. Since the Trident EVF information model associates the workflow instances of a given version of the workflow, scientists can even see the runtime information of each and every workflow execution. We believe that providing this information along a time-line will give users more insight into their research process.

# 4 Related Work

Provenance in scientific workflows has received attention in the recent past. Several workflow systems support provenance recording, such as Taverna [16] and Kepler [15]. Taverna [16] uses a pure dataflow model for its workflows and its provenance capture is also limited to dataflow provenance. Trident in addition captures the control flow aspects of the workflow and its provenance. Taverna also captures provenance for hierarchical data collections, which Trident does not.

There are also stand alone provenance tracking and storage systems like Karma [11] and PASOA [7]. Karma [11] uses instrumentation of workflow engine and Axis2 web service activities for recording provenance in a database, also using a pub-sub model for event transfer. It has been demonstrated with G-BPEL and ODE workflow engines. Trident uses similar instrumentation but is more tightly integrated with the Windows WF engine and activities. This is partly due to the reuse of existing tracking information provided by Windows WF, and enables features like tracking provenance of interactive user operations on Windows GUI elements like graphs and forms. Karma and PASOA support dual views of provenance tracking, both from the workflow engine and activity/web service. Trident does not make this distinction. However, neither libraries support transparent use of diverse backend storage such as file, database, and cloud that the Trident Registry enables. Despite these differences, there is overlap on the provenance collected by the different provenance systems as demonstrated in the third Provenance Challenge workshop, where provenance data collected in Trident was able to interoperate with the provenance systems of Taverna, Kepler, Karma and PASOA [19].

# 5 Summary

Scientific workflows have emerged as the de facto model for researchers to process, transform and analyze scientific data. Workflow management systems provide researchers with many valuable and time saving features, from cataloging workflows, workflow activities and web services, to visual authoring and workflow monitoring. But arguably the most valuable service they offer is the automatic capture of provenance data sufficient to establish trust in a result and potentially allow other researchers to reproduce a result. In this paper we have summarized highlights of the workflow provenance that is automatically collected and managed by the Trident Scientific Workflow Workbench. Trident provides an integrated way to collect, store and view provenance for workflows, and supports a range of provenance queries over workflow results. The implementation is based on the ability to intercept and routes low level events through a scalable and high performance pub-sub API.

In addition Trident tracks workflow evolution provenance to record changes to individual workflows over time and correlates versioned workflows with the results they generate. The provenance data collected by Trident is compatible with the emerging Open Provenance Model standard and different data stores are supported for storing provenance.

## References


[1] "Fluxnet," http://www.fluxdata.org.

[2] "Large hadron collider (lhc)," http://public.web.cern.ch/public/en/LHC/LHC-en.html.

[3] "Large synoptic sky survey (lsst)," http://www.lsst.org/lsst_home.shtml.

[4] "Nih public access policy," http://publicaccess.nih.gov/policy.htm.

[5] "Project trident download," http://research.microsoft.com/en-us/collaboration/tools/trident.aspx.

[6] "Windows workflow foundation (winwf)," http://en.wikipedia.org/wiki/Windows_Workflow_Foundation.

[7] *Recording and Using Provenance in a Protein Compressibility Experiment*, 2005.

[8] *Versioning for Workflow Evolution*, Chicago, IL, June 2010. [Online]. Available: http://www.cct.lsu.edu/~kosar/didc10/papers/didc06.pdf

[9] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan, "The trident scientific workflow workbench," *eScience, IEEE International Conference on*, vol. 0, pp. 317–318, 2008.

[10] D. Box and A. Hejlsberg, "Linq: .net language-integrated query," http://msdn.microsoft.com/en-us/library/bb308959.aspx.

[11] B. Cao, B. Plale, G. Subramanian, E. Robertson, and Y. Simmhan, "Provenance information model of karma version 3," in *IEEE 2009 Third International Workshop on Scientific Workflows*, 2009.

[12] C. A. Goble and D. C. De Roure, "myexperiment: social networking for workflow-using e-scientists," in *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*. New York, NY, USA: ACM, 2007, pp. 1–2.

[13] T. Hey, S. Tansley, and K. Tolle, "The fourth paradigm: data-intensive scientific discovery," 2009, http://research.microsoft.com/en-us/collaboration/fourthparadigm/.

[14] L. Lins, D. Koop, E. Anderson, S. Callahan, E. Santos, C. Scheidegger, J. Freire, and C. Silva, "Examining statistics of workflow evolution provenance: A first study," in *Scientific and Statistical Database Management*. Springer, 2008, pp. 573–579.

[15] T. McPhillips, S. Bowers, D. Zinn, and B. Ludscher, "Scientific workflow design for mere mortals," *Future Generation Computing Systems*, 2009.

[16] P. Missier, S. Sahoo, J. Zhao, C. Goble, and A. Sheth, "Janus: from workflows to semantic provenance and linked open data."

[17] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, and P. Paulson, "The open provenance model," *University of Southampton*, 2007.

[18] Y. Simmhan, C. van Ingen, A. Szalay, R. Barga, and J. Heasley, "Building reliable data pipelines for managing community data using scientific workflows," in *e-Science, 2009. e-Science '09. Fifth IEEE International Conference on*, 9-11 2009, pp. 321 –328.

[19] Y. Simmhan and R. Barga, "Analysis of approaches to supporting the open provenance model in the trident workflow workbench," *Future Generation Computing Systems (in Review)*, 2010.

[20] Y. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *ACM SIGMOD Record*, vol. 34, no. 3, p. 36, 2005.

[21] M. Valerio, S. Sahoo, R. Barga, and J. Jackson, "Capturing Workflow Event Data for Monitoring, Performance Analysis, and Management of Scientific Workflows," in *IEEE Fourth International Conference on eScience, 2008. eScience'08*, 2008, pp. 626–633.

# Causality in Databases*

Alexandra Meliou[1], Wolfgang Gatterbauer[1], Joseph Y. Halpern[2], Christoph Koch[2], Katherine F. Moore[1], and Dan Suciu[1]

[1]University of Washington
{ameli,gatter,kfm,suciu}@cs.washington.edu

[2]Cornell University
{halpern,koch}@cs.cornell.edu

**Abstract**

*Provenance is often used to validate data, by verifying its origin and explaining its derivation. When searching for "causes" of tuples in the query results or in general observations, the analysis of lineage becomes an essential tool for providing such justifications. However, lineage can quickly grow very large, limiting its immediate use for providing intuitive explanations to the user. The formal notion of* causality *is a more refined concept that identifies causes for observations based on user-defined criteria, and that assigns to them gradual degrees of responsibility based on their respective contributions. In this paper, we initiate a discussion on causality in databases, give some simple definitions, and motivate this formalism through a number of example applications.*

## 1 Introduction

The notion of causality and causation is a topic in philosophy, studied and argued over by philosophers over the centuries. A formal, mathematical study of causation and responsibility has been initiated recently by the work of Pearl [Pea00] and Halpern and Pearl [HP05], who have distilled the generally accepted aspects of causality into a rigorous definition. This work has been influential in the Computer Science research community and has already lead to applications in model checking and verification [BBDC$^+$09, CHK08, CGY08].

The goal of this paper is to initiate a discussion on causality in databases. In Sec. 2, we give a minimalistic definition of query causality that still captures the key concepts in the HP definition: (1) counterfactual v.s. actual cause, (2) the notion of a contingency set, (3) Chockler and Halpern's notion of responsibility [CH04]. We also show its relation to provenance, also refered to as *lineage* in this paper. Then in Sec. 3, we show that our definition is general enough to be applied to a broad class of database-related applications.

## 2 Definitions

Causality in databases aims to answer the following question: *given a query over a database instance and a particular output of the query, which tuple(s) in the instance caused that output to the query?* An important

*Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

*This work was partially supported by NSF III 0911036.

contribution of Halpern and Pearl's definition of causality is its reliance on *structural equations*, which express the notion of *intervention*: what happens to the outcome if we change the state of the input. As a starting point, we propose to define causality in databases by using a single structural equation: the lineage of that answer to the query. We first define causality without referring to lineage, then relate the two notions.

## 2.1 Causality

Following Halpern and Pearl (HP from now on), we partition the tuples in the database into *exogenous* and *endogenous* tuples. The endogenous tuples are those that are considered candidates for causes. Exogenous tuples are deemed not to be possible causes: they define the context determined by external factors. This partition depends on the particular application and is defined either by the system or by the user. Let $D^{\mathrm{n}} \subseteq D$ denote the *endogenous tuples*; the *exogenous tuples* are then $D^{\mathrm{x}} = D - D^{\mathrm{n}}$. For each relation name $R_i$, we write $R_i^{\mathrm{n}}$ and $R_i^{\mathrm{x}}$ to denote the endogenous and exogenous tuples in $R_i$, respectively. Thus, $R_i = R_i^{\mathrm{n}} \cup R_i^{\mathrm{x}}$.

Our definition of causality simplifies the HP definition while still retaining its two key elements, *counterfactual causes* and *contingencies*.

**Definition 1 (DB Causality):** Let $t \in D^{\mathrm{n}}$ be an endogenous tuple, and $r$ a possible answer to query $q$.

- $t$ is called a *counterfactual cause* for $r$ in $D$ if $D \models q(r)$ and $D - \{t\} \not\models q(r)$.
- $t \in D$ is called an *actual cause* for $r$ if there exists a set $\Gamma \subseteq D^{\mathrm{n}}$, called a *contingency* for $t$, such that $t$ is a counterfactual cause for $r$ in $D - \Gamma$.

A tuple $t$ is a counterfactual cause if its removal from the database also removes $r$ from the query result. The tuple is an actual cause if one can find a contingency for which the tuple becomes a counterfactual cause: more precisely, one has to find a set of tuples $\Gamma$ such that, after removing $\Gamma$ from the database we bring the database to a state where removing/inserting $t$ causes $r$ to switch between an answer and a non-answer. The set $\Gamma$ is called a contingency for $t$. Obviously, every counterfactual cause is also an actual cause, by taking $\Gamma = \emptyset$.

Our simplified definition of causality is analogous to the definition for causality in Boolean circuits used in [CHK08]. For monotone queries, it can be shown that this simplified definition agrees with the full HP definition. For example, one requirement that we dropped is the following restriction on the contingency $\Gamma$: for any $\Gamma' \subseteq \Gamma$, it is required that $D - \Gamma' \models q(r)$. In other words, if we keep the candidate cause $t$ in the database, then any subset of the contingency should not change the output to the query. If the query is monotone then this condition is automatically satisfied by our simple definition (as $D - \Gamma \models q(r)$ implies $D - \Gamma' \models q(r)$). However, for non-monotone queries, our simple definition differs from the HP definition and may be inadequate. For example, consider a non-monotone query over a database containing three tuples $t_1, t_2, t$. The query evaluates to true if both $t_1$ and $t_2$ are in the database, or if $t$ is in the database and neither of $t_1, t_2$ are; in other words, its lineage is $(X_{t_1} \wedge X_{t_2}) \vee (\neg X_{t_1} \wedge \neg X_{t_2} \wedge X_t)$ (see Sec. 2.2). In our simplified definition, $t$ is an actual cause with contingency $\Gamma = \{t_1, t_2\}$ (as removing $t_1, t_2$ from the database makes $t$ counterfactual). Under the full HP definition, this contingency is not allowed because the query becomes false by just removing $t_1$ from the database. For a preliminary attempt towards a more general adaptation of the HP definition in a database context, see [MGMS09].

## 2.2 Relating Causality and Lineage

We can now relate causality and lineage. We define lineage as a Boolean expression, specifically based on the Boolean c-tables representation [GT06, GKT07]. Associate a Boolean variable $X_t$ to each tuple $t$ in the database instance $D$. The *lineage* of an answer $r$ to a query $q$ is defined as the unique (up to logical equivalence) Boolean expression $\varphi^{q,r,D}$ such that, for every truth assignment $\theta$ of $X_t$, $\varphi^{q,r,D}[\theta] = \texttt{true}$ iff the query $q$ returns the answer $r$ when restricted to the subset $D_0 = \{t \mid \theta(X_t) = \texttt{true}\} \subseteq D$. For example, the lineage of the answer $c$ to the query $q(x) :- R(x, y), S(y)$ might be $(X_{R(c,b)} \wedge X_{S(b)}) \vee (X_{R(c,f)} \wedge X_{S(f)})$, which says that $c$ will be an answer to $q$ on any subset of the database that contains the tuples $R(c, b)$ and $S(b)$, or the tuples $R(c, f)$ and $S(f)$.

The lineage expression gives us a simple tool for computing all causes for an answer $r$ to a monotone query $q$. Write the lineage $\varphi^{q,r,D}$ in DNF, then substitute all Boolean variables corresponding to exogenous variables with `true`; call the result $\psi$. A *minterm* of $\psi$ is a minimal set of tuples $T$ such that $\bigwedge_{t\in T} X_t \Rightarrow \psi$. Then every tuple that occurs in a minterm of $\psi$ is a cause for the answer $r$.

**Proposition 2 (Lineage Minterms):** Let $\varphi^{q,r,D}$ be the lineage expression for an answer $r$ to a monotone query $q$ over a database instance $D$. Let $\psi^{q,r,D}$ be obtained from $\varphi^{q,r,D}$ by replacing all Boolean variables for exogenous tuples with `true`. Then tuple $t$ is an actual cause of the answer $r$ to the query $q$ on $D$ iff there exists a minterm in $\psi^{q,r,D}$ that contains $t$.

For example, if $\psi = (X_{t_1} \wedge X_{t_2} \wedge X_{t_3}) \vee (X_{t_1} \wedge X_{t_3} \wedge X_{t_4}) \vee (X_{t_1} \wedge X_{t_2} \wedge X_{t_3} \wedge X_{t_5})$, then there are two minterms, $\{t_1, t_2, t_3\}$ and $\{t_1, t_3, t_4\}$, and the actual causes are $t_1, t_2, t_3, t_4$; the tuple $t_5$ is not a cause because the third term in $\psi$ is redundant. In the case where all tuples are endogenous, Proposition 2 eliminates terms that do not appear in the *minimal witness basis* of the answer [BKT01], and therefore the set of actual causes is equivalent to the union of all sets in the minimal witness basis.

### 2.3 Degree of Responsibility

To measure the degree to which a tuple is considered a cause, Chockler and Halpern [CH04] introduced the notion of *degree of responsibility*, or simply *responsibility*. We present the intuition with a simple example: Assume 11 voters who can vote for either candidate A or candidate B. In an 11-0 outcome, each voter is a cause for A's win, but each one's responsibility is less than in a 6-5 outcome. The following definition, an adaptation from Chockler and Halpern [CH04], captures this intuition.

**Definition 3 (DB Responsibility):** Let $r$ be an answer to a query $q$, and let $t$ be a cause. With $\Gamma$ ranging over all possible contingencies for $t$, the *responsibility* of $t$ for the answer $r$ is $\rho_t = (1 + \min_\Gamma |\Gamma|)^{-1}$

Thus, the responsibility of $t$ is a function of the minimal number of tuples that we need to remove from the real database $D$ before $t$ becomes counterfactual. The tuple $t$ is a counterfactual cause iff $\rho_t = 1$. If $t$ is not a cause then, by convention, we take $\rho_t = 0$. Intuitively, the larger the contingency, the less counterfactual a variable is, and thus its responsibility is lower. In the case of the 6-5 vote, each vote for A is critical, so it has degree of responsibility 1. On the other hand, in the case of the 11-0 vote, each vote has degree of responsibility $1/6$; 5 votes have to be changed before one becomes critical. Some first results on the complexity of computing causality and responsibility for the case of conjunctive queries can be found in [MGMS11].

**Constraints.** The definitions of causality and responsibility should be revisited in the presence of known constraints over the data. Take for example $q(x) :- R(x, y, z), S(x, y, u)$, with the constraint $R(x, y) \subseteq S(x, y)$. Under this constraint, $q$ is equivalent to $q'(x) :- R(x, y, z)$, and yet computing responsibility over $q$ and $q'$ would yield different results. Constraints are beyond the scope of this paper, but important to consider in future work.

## 3 Applications of Causality and Responsibility in Databases

### 3.1 Explaining Unexpected Answers

We start with the most basic application: explaining unexpected answers to a query. Consider the IMDB movie database, available from `www.imdb.org`. It consists of the six tables illustrated in Fig. 1. Tim Burton is an Oscar-nominated director well known for fantasy movies involving dark, Gothic themes. Examples of his work include "Edward Scissorhands", "Beetlejuice", and the recent "Alice in Wonderland". A user wishing to learn more about Burton's movies queries the IMDB dataset to find out about the genres of movies that he has directed; the query issued, shown in Fig. 1, retrieves all genres of movies directed by Burton. The query's answer, ran on the actual database and shown in Fig. 1, includes some expected genres, such as Fantasy and Horror, and

**Database Schema**

Director(*did*, *firstName*, *lastName*)
Actor(*aid*, *firstName*, *lastName*)
Movie(*mid*, *name*, *year*, *rank*)
Movie_Directors(*did*, *mid*)
Genre(*mid*, *genre*)
Casts(*aid*, *mid*, *role*)

**Query**

```
select    distinct g.genre
from      Director d, Movie_Directors md,
          Movie m, Genre g
where     d.lastName like 'Burton'
          and g. mid=m.mid
          and m. mid=md.mid
          and md. did=d.did
order by  g.genre
```

**Query Answers**

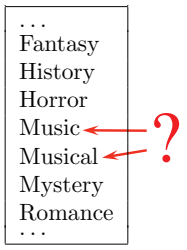



Figure 1: Example constructed with real data from the IMDB dataset. The user issues the query above to identify the movie genres that Burton has directed, and is surprised to see the categories `Music` and `Musical` listed. (Note that a movie may be classified under several genres.)

some genres that seem more suspicious, such as Music and Musical. Are the latter errors in the data or the query, or are they interesting discoveries? In other words, what are the causes of Musical and Music appearing in the query's answer?

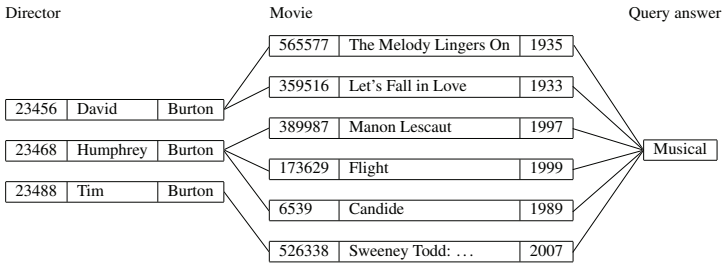
Tracing back the lineage of each answer to the query is a first step towards explaining them, but it is not sufficient. In combination, the lineage of these two answers consists of a total of 137 base tuples, and this, we argue, is overwhelming to a user. Representing this lineage as a list of minterms (44 in this case), still requires further analysis by the user to determine the frequency of a tuple's appearance in the lineage, as well as its participation in minterms of possibly different sizes, to determine its actual significance to the result. It also turns out that all 137 tuples are actual causes (by Prop. 2), so causality does not provide us with any more information than lineage in this case. However, ranking by responsibility gives us much more interesting information, as we now show.

Consider the case of the answer Musical, whose lineage is much smaller than Music and shown in Fig. 2a. [1] We can see immediately the causes of the suspicious answer. Although Tim Burton *did* direct one musical, "Sweeney Todd" (surprise!), there are two other directors with last name Burton (oops!) who directed several musicals. What we would like from a system is to list "Sweeney Todd", David Burton, and Humphrey Burton as top causes for Musical: the first is a surprising finding, the last two point to a bug in the query. We don't want a system to list a movie like "The Melody Lingers On" as a top explanation, because that doesn't really explain directly the surprising answer Musical. Ranking by responsibility achieves precisely this behavior, as shown in Fig. 2b. The movie "Sweeney Todd" has a minimum contingency of size two, consisting of the directors David and Humphrey Burton, because by removing these two directors from the database, "Sweeney Todd" becomes counterfactual. In addition, each of the three directors with last name Burton also has a contingency of size two, consisting of the other two directors. Thus, these four tuples (the movie and the three directors) have the same responsibility, $1/3$, and are listed as the top explanations for why Musical is returned by the query. On the other hand, the smallest contingency for the movie "Manon Lescaut" has size four,[2] hence its responsibility is 1/5. The answer Music has a much larger lineage consisting of 118 tuples; ranking them by responsibility proves to be a key technique in identifying interesting explanations. In this case Tim Burton and his movie "Corpse Bride" appear highest in the ranking, along with three more directors who share the same last name: Al, Tom, and Humphrey.

**Discussion.** Our example illustrates a general phenomenon. Tuples with high responsibility tend to be interesting explanations to query answers; tuples with low responsibility tend to be uninteresting. The reason for why a tuple is interesting may vary: it may be a surprising fact in the data, it may point to a bug in the query, or it may represent an error in the data. Our simple definition of causality does not distinguish between these explanations, but only ranks them by the degree of interestingness to the user. A challenging research direction

[1] The lineage has 21 tuples. Tuples from the `Movie_Directors` and `Genre` relations are depicted in Fig. 2a as edges between the `Director` and `Movie` tuples and the `Movie` and answer tuples, respectively. It is natural to consider the latter exogenous; if they were endogenous, they would have the same responsibility as the movie they correspond to.

[2] {`Movie`("Candide"), `Movie`("Flight"), `Director`(David Burton), `Director`(Tim Burton)}.




(a) Lineage from `Directors` and `Movies` of the Musical tuple

| Answer tuple | $\rho_t$ |
|---|---|
| Movie(526338, "Sweeney Todd", 2007) | 0.33 |
| Director(23456, David, Burton) | 0.33 |
| Director(23468, Humphrey, Burton) | 0.33 |
| Director(23488, Tim, Burton) | 0.33 |
| Movie(359516, "Let's Fall in Love", 1933) | 0.25 |
| Movie(565577, "The Melody Lingers On", 1935) | 0.25 |
| Movie(6539, "Candide", 1989) | 0.20 |
| Movie(173629, "Flight", 1999) | 0.20 |
| Movie(389987, "Manon Lescaut", 1997) | 0.20 |

(b) Responsibility rankings for Musical

Figure 2: Ranking causes in the IMDB example. Displayed are the results for the Musical tuple, assuming that all the tuples in the relations `Directors` and `Movies` are endogenous.

would be to refine these notions in order to adapt them to specific purposes.

### 3.2 Diagnosing Network Failures

With the basic intuition in mind, we will illustrate now a quite different application of causality. Consider a computer network consisting of a number of servers, with physical links between the servers stored in a table `Link(srv1,srv2)`. At any given time, a server may be active or down, and `Active(Srv)` is the table of servers that are assumed to be active. A network administrator, call her Alice, is concerned in maintaining the network's connectivity, which can be expressed by the following Datalog program:

```
Connected(x,y) :- Link(x,y),Active(x),Active(y)
Connected(x,y) :- Connected(x,z),Link(z,y),Active(y)
Connected(x,y) :- Connected(y,x)
```

The last rule expresses the fact that the graph is undirected. If a server is down and Alice knows that, she will take immediate corrective action, by rebooting the server or replacing the physical machine. Thus, under normal operation, all servers are up and the network is connected.

At any given time, `Active` contains all servers that are believed by Alice to be active. Suppose that, at some point, Alice observes that a server `A` is no longer connected to a server `B`. She can learn this from failed processes, or by examining logs, or by receiving a system alert. Some servers in the network are down, and this caused `A` and `B` to become disconnected. Alice's challenge is to identify which servers are down. Thus, the content of `Active` is no longer up to date: if we run the query `Connected`, it still returns the answer `A,B`, but this clearly disagrees with the actual observation because `Active` is not up to date. Here we use causality and responsibility to help diagnose the real system, instead of the database. Suppose that we declare all tuples in `Link` to be exogenous; only tuples in `Active` are endogenous. In other words, we assume that links do not fail, only servers. Then the lineage of the answer `Connected(A,B)` consists of all active servers on all simple paths connecting `A` to `B`, and each server `C` on a simple path from `A` to `B` is an actual cause. Ranking these servers by responsibility will help Alice diagnose the network. Indeed, a minimal contingency for a server `C` to be a cause is a set of servers $\Gamma$ such that $\Gamma \cup \{\texttt{C}\}$ is a minimal cut disconnecting `A` from `B`. Thus, servers with a high responsibility are a more likely explanation for the observed disconnection in the network, while servers with low responsibility are less likely explanations. By ranking servers in decreasing order of their responsibility the system assists the network administrator with valuable information, allowing her to prioritize her diagnosis work.

**Extensions.** In a well-studied problem called *network reliability* [Col87], we are given a network where each edge (or node) has a given probability of failing, and are asked to compute the probability that the network remains connected. The network diagnosis example is clearly related to network reliability, and this connection suggests the following extension of causality and responsibility. Suppose that for each tuple in the database we are given a score (or weight): then it makes sense to define the responsibility of a tuple $t$ as a function of the mini-

mum weight of a contingency $\Gamma$ for $t$. Thus, we no longer consider only the cardinality of $\Gamma$, but take into account the weights of the tuples in $\Gamma$. In the case of network reliability, the weight of a node $s$ can be the probability that $s$ fails, that is, $s$ is active with probability $1-p(s)$, and the probability of a contingency $\Gamma$ is $\prod_{s\in\Gamma} p(s) \prod_{s\notin\Gamma}(1-p(s))$. Define the responsibility of a tuple $t$ to be the maximum probability of all contingencies.

Halpern [Hal08] has recently discussed a related refinement of causality, by combining it with default reasoning. In that setting, the weight of a world represents how normal that world is. $\mathcal{K}(D_1) \leq \mathcal{K}(D_2)$ means that $D_1$ is more "normal" than $D_2$, where $\mathcal{K}$, is a given ranking function; equivalently, $D_2$ is more "exceptional" than $D_1$). The definition of causality is then modified by requiring the contingency $\Gamma$ to lead to a more "normal" world, that is, $\mathcal{K}(D-\Gamma) \leq \mathcal{K}(D)$. Notice that this is different from changing the definition of responsibility, because it restricts the set of allowable contingencies rather than modifying their numerical value. The two extensions are motivated by different applications. In the case of network reliability, *every* contingency containing at least one failure is more exceptional than the normal state (with no failures), so restricting contingencies does not make a difference here.

### 3.3 Handling Aggregate Queries

Causality poses additional challenges when the query contains aggregates. Consider a simple aggregate query: `select sum(A) from R`. Here every tuple in `R` that has `R.A`$\neq$`0` is counterfactual, because its addition/removal changes the sum, but obviously not all tuples in `R` are equal contributors to the sum. This highlights the limitations of our simple definition of causality. It would be interesting to extend the definition of causality and responsibility to quantify the contribution of each input tuple to the aggregate, which may be challenging when the query combines aggregates with joins.

However, even our simple definition of causality captures some interesting aspect of aggregate queries. Consider the question "why is the sum greater than 500?". This can be phrased as a causality question for the boolean query $Q$: `select 'true' from R having sum(A) > 500`, to which our notion of causality and responsibility applies. Assume the following instance of $R.A$: $\{450, 150, 75, 25\}$. In this case, $450$ is a counterfactual tuple for $Q$ (removing it makes $Q$ false); $150$ is an actual cause with contingency $\{75\}$, while $25$ is not an actual cause at all. The responsibilities are $\rho_{450} = 1$, $\rho_{150} = \rho_{75} = 1/2$, and $\rho_{25} = 0$ (for details see [MGMS09]).

Typical monitoring queries contain several joins and aggregates, and generate alerts based on whether an aggregate reached a specified value, that is, `having count(*) > N`. Causality aims to answer the question "why did the number of events (or errors, or failures) exceed $N$?". We note here that, while causality and responsibility are captured by our definition, the computational challenges may be much higher in the case of queries with aggregates, because the lineage may have a size that is exponential in $N$, and therefore it is not possible to compute explicitly. New algorithms are likely to be needed in order to efficiently compute or approximate the responsibility of tuples in queries with aggregates.

### 3.4 View-Conditioned Causality

In all previous examples we have assumed that there is a single query $q$, and a single answer $r$ to that query, and have defined causality for a fixed $q$ and $r$. In practice one often has multiple queries and/or multiple answers. Fix a query $q$ and an answer $r$, and consider $k$ additional queries, called views, $v_1, \ldots, v_k$, with answers $b_1, \ldots, b_k$. The answers to the views are normal and expected, but the answer to the query is unexpected. We want to explain the cause for the answer $r$ to $q$, conditioned on the answers to the views: we call this *View-Conditioned causality*.

We define view-conditioned causality by making the following small change to Definition 1: we require a contingency $\Gamma$ to be such that $D - \Gamma - \{t\} \models v_i(b_i)$, for all $i = 1, k$. This is related to the view side-effect problem studied by Buneman et al. [BKT02], which asks for changes to the database that have minimal side-effects on some given views. We illustrate the usefulness of this notion on the examples considered so far:

- Suppose that all answers in Fig. 1 are normal except Music and Musical. With the revised definition, we consider as contingencies only those sets of tuples whose removal does not affect the other query answers.
- Consider a monitoring query that is run periodically on a data stream: for example the query may count the number of dropped IP packets per minute in a computer network. The query returns a stream of answers, $a_1, a_2, \ldots, a_{t-1}, a_t$. Suppose that the last answer $a_t$ is abnormal: there is a spike in the number of dropped packets. The natural question to ask is: what is the cause for the spike $a_t$ given that the previous answers $a_1, a_2, \ldots, a_{t-1}$ were normal? This is captured by view-conditioned causality, where $a_1, \ldots, a_{t-1}$ are the views and $a_t$ is the query.
- Consider a distributed query execution environment, such as Pig over Hadoop [GNC+09, ORS+08]. Complex Pig (or Map-reduce) jobs take hours to run, and their performance depends on a large number of parameters, such as the number of servers, the size of the data, number of reduce tasks, size of the data chunks, etc. Suppose that the same job is run repeatedly, with slightly different parameter settings, and results in running times $a_1, a_2, \ldots, a_n$. Suddenly, the last running time $a_n$ is abnormally high. The interesting question is: what caused $a_n$ to be abnormally high, given that $a_1, a_2, \ldots, a_{n-1}$ were normal?
- Assume the network administrator (Alice) has more information about the actual state of the network: she knows that `C` and `D` are still connected, and so are `E` and `F`, but that `A,B` are disconnected and so is the pair `M,N`. View-conditioned causality allows her to ask the question: why are certain pairs of servers disconnected while others remain connected?

**Discussion.** Conditioning on views seems to be a key ingredient to causality in databases. We can think of exogenous tuples as those we want to take as given; view-conditioning allows us to to extend this intuition by treating tuples in the view as given.

### 3.5 Preemption

A major effort in the original HP definition of causality consisted in capturing correctly the notion of preemption, which had been discussed in philosophy and turned out to be difficult to capture in a mathematical formalism. Preemption occurs when one possible cause is known to rule out another possible cause. The standard example of preemption given in [HP05] and elsewhere is when Alice and Bob both throw a rock at a bottle: both aim accurately, but Alice throws first, and therefore her throw hits the bottle and is the cause the bottle breaking, not Bob's. Here preemption is due to temporal ordering, where events that happened earlier preempt events that happened later. For another example, suppose one wants to trace the source of an infection with a computer virus: servers that were infected earlier preempt (as potential causes) servers that were infected later. Non-temporal criteria may also be reasons for preemption. For example, suppose that a burst in stock purchases caused a certain stock to jump 5 points in a few minutes: when searching for the causes of the sudden increase, transactions with high volume should preempt transactions of low volume, as their contribution to the change is higher.

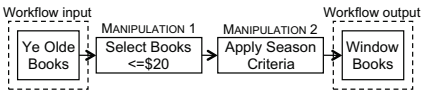
The HP definition approach deals with preemption by choosing the appropriate structural equations. If both Alice and Bob throw a rock at a bottle and we take the equation for bottle breaks ($BB$) to be $BB = A \vee B$ (where $A$ is "Alice throws" and $B$ is "Bob throws"), then both $A$ and $B$ are actual causes, because the set $\{B\}$ is a contingency for $A$ and vice versa. This equation treats $A$ and $B$ symmetrically; it does not capture the fact that Alice's throw hit the bottle before Bob's did. It is beyond the scope of this paper to review Halpern and Pearl's full definition of causality and responsibility and explain how it addresses preemption. The HP treatment of preemption seems to require the more complicated full HP definition. A major challenge is that of finding a definition more in the spirit of the one used in this paper which handles preemption well.

### 3.6 Why-not Causality for non-answers

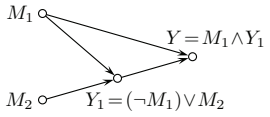
Explanation of query answers naturally extends to the absence of expected results. Recent work has focused on the *why-not* question, that is, *why a certain tuple is not in the result set*. The problem has been addressed from

| Author | Title | Price | Publisher |
|---|---|---|---|
| | Epic of Gilgamesh | $150 | Hesperus |
| Euripides | Medea | $16 | Free Press |
| Homer | Iliad | $18 | Penguin |
| Homer | Odyssey | $49 | Vintage |
| Hrotsvit | Basilius | $20 | Harper |
| Longfellow | Wreck of the Hesperus | $89 | Penguin |
| Shakespeare | Coriolanus | $70 | Penguin |
| Sophocles | Antigone | $48 | Free Press |
| Virgil | Aeneid | $92 | Vintage |

(a)




(b)




(c)

Figure 3: Example book store database from [CJ09] (a), representation of the workflow (b), and the corresponding causal network (c).

both the perspectives of data [HCDN08], which is our focus, and query revisions [CJ09, TC10]. Causality unifies explanations for answers and non-answers under a common framework. We demonstrate how causality, with the notion of preemption discussed in Sec. 3.5, can model an example presented by Chapman and Jagadish in [CJ09].

**Example 1 (Book Shopper (Ex. 1) [CJ09]):** A shopper knows that all "window display books" at Ye Olde Booke Shoppe are around $20, and wishes to make a cheap purchase. She issues the query: Show me all window display books. Suppose the result from this query is (Euripides, Medea). Why is (Hrotsvit, Basilius) not in the result set? Is it not a book in the store? Does it cost more than $20? Is there a bug in the query-database interface such that the query was not correctly translated?

Chapman and Jagadish consider a discrete component of a workflow, called *manipulation*, as an explanation of a why-not query. The workflow describing the query of the example is shown in Fig. 3b. Roughly, a manipulation is considered *picky* for a non-result if it prunes the tuple. For example, manipulation 1 of Fig. 3b is picky for "Odyssey", as it costs more than $20. Equivalently, a manipulation is *frontier picky* for a set of non-results if it is the last in the workflow to reject tuples from the set. In this framework, the cause of a non-answer will be a frontier picky manipulation.

In Example 1, tuple $t =$(Hrotsvit, Basilius) passes the price test, but is rejected by manipulation 2 as it does not satisfy the seasonal criteria. The causal network representing this example is presented in Fig. 3c. Input nodes model the events and describe the pickiness of manipulations: $M_1$ holds if manipulation 1 is *not* potentially picky with respect to $t$, and $M_2$ holds if manipulation 2 is *not* potentially picky with respect to $t$. At the end, the tuple appears only if neither manipulation is picky: $M_1 \wedge M_2$. Node $Y_1$ encodes the precedence of the manipulations in the workflow. A tuple will be stopped at point $Y_1$ of the workflow if $M_2$ is picky but $M_1$ was not, that is, if $M_1 \wedge \neg M_2$ holds. Thus, it will pass this point if $Y_1 = \neg(M_1 \wedge \neg M_2) = \neg M_1 \vee M_2$ holds. A tuple is reported if $Y = M_1 \wedge Y_1$ holds.

Applying the HP framework in the context of the example where $M_2$ is picky but $M_1$ is not ($M_1 \wedge \neg M_2$), correctly yields that $M_2$ is the only cause (counterfactual): $\Gamma = \emptyset$. If both manipulations were potentially picky ($\neg M_1 \wedge \neg M_2$), the HP definition again correctly picks $M_1$ as the only cause with contingency $\Gamma = \{M_2\}$. Hence, even though $M_2$ is potentially picky, it is preempted by $M_1$. This is in agreement with the why-not framework that selects as explanation the last manipulation in the workflow that finally rejected the tuple.

This example demonstrates not only that causality can model why-not questions, but also that it can handle as causes any element that may be considered contributory to an event (in this case workflow manipulations). However, when focusing on tuples, we can again use a simple definition along the lines of the one presented in Sec. 2, this time tweaked to specify causes and contingencies for missing tuples. This requires some minor changes to Definition 1. Now the real database consists entirely of exogenous tuples $D^{\mathrm{x}}$. In addition, we are given a set of potentially missing tuples, whose absence from the database caused $r$ to be a non-answer: these form the endogenous tuples $D^{\mathrm{n}}$, and we denote $D = D^{\mathrm{x}} \cup D^{\mathrm{n}}$. We do not discuss here how to compute the set

$D^{\mathrm{n}}$; we assume that $D^{\mathrm{n}}$ is given (one possibility is to compute it using techniques from [HCDN08]). Definition 1 is then modified as follows:

**Definition 4 (Why-not Causality):** Let $t \in D^{\mathrm{n}}$ be an endogenous tuple, and $r$ a missing answer for $q$.

- $t$ is called a *counterfactual cause* for the non-answer $r$ in $D^{\mathrm{x}}$ if $D^{\mathrm{x}} \not\models q(r)$ and $D^{\mathrm{x}} \cup \{t\} \models q(r)$
- $t \in D$ is called an *actual cause* for the non-answer $r$ if there exists a set $\Gamma \subseteq D^{\mathrm{n}}$ such that $t$ is a counterfactual cause for the non-answer of $r$ in $D^{\mathrm{x}} \cup \Gamma$.

# 4 Conclusions and Future Work

This paper initiates a discussion of causality in databases. It gives a simple definition of causality in database queries, and illustrates it with several applications, including explanations, validation, data correction, and system debugging. Currently, such tasks can be performed by manually exploring provenance information. *Causality* allows to refine the analysis of provenance by including user and application specific constraints, and provides meaningful ranking metrics. Future work needs to specialize this definition to more concrete applications, and to study the computational challenges associated with computing cause and responsibility.

# References


[BBDC+09] Ilan Beer, Shoham Ben-David, Hana Chockler, Avigail Orni, and Richard J. Trefler. Explaining counterexamples using causality. In *21st Annual Conference on Computer Aided Verification (CAV '09)*, pages 94–108, 2009.

[BKT01] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[BKT02] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. On propagation of deletions and annotations through views. In *PODS*, pages 150–158, 2002.

[CGY08] Hana Chockler, Orna Grumberg, and Avi Yadgar. Efficient automatic STE refinement using responsibility. In *Proc. 14th Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 233–248, 2008.

[CH04] Hana Chockler and Joseph Y. Halpern. Responsibility and blame: A structural-model approach. *J. Artif. Intell. Res. (JAIR)*, 22:93–115, 2004.

[CHK08] Hana Chockler, Joseph Y. Halpern, and Orna Kupferman. What causes a system to satisfy a specification? *ACM Trans. Comput. Log.*, 9(3), 2008.

[CJ09] Adriane Chapman and H. V. Jagadish. Why not? In *SIGMOD*, pages 523–534, 2009.

[Col87] Charles J. Colbourn. *The combinatorics of network reliability*. Oxford University Press, New York, 1987.

[GKT07] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 271–280, 2007.

[GNC+09] Alan Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan Narayanam, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, and Utkarsh Srivastava. Building a highlevel dataflow system on top of MapReduce: The pig experience. *PVLDB*, 2(2):1414–1425, 2009.

[GT06] Todd J. Green and Val Tannen. Models for incomplete and probabilistic information. *IEEE Data Eng. Bull.*, 29(1):17–24, 2006.

[Hal08] Joseph Y. Halpern. Defaults and normality in causal structures. In *KR*, pages 198–208, 2008.

[HCDN08] Jiansheng Huang, Ting Chen, AnHai Doan, and Jeffrey F. Naughton. On the provenance of non-answers to queries over extracted data. *PVLDB*, 1(1):736–747, 2008.

[HP05] Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *Brit. J. Phil. Sci.*, 56:843–887, 2005. (Conference version in *UAI*, pages 194-202, 2001).

[MGMS09] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. Why so? or Why no? Functional causality for explaining query answers. *CoRR*, abs/0912.5340, 2009.

[MGMS11] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. The causality and responsibility of query answers and non-answers. In *PVLDB*, 2011. (to appear, see `http://db.cs.washington.edu/causality/`).

[ORS+08] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD*, pages 1099–1110, 2008.

[Pea00] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, Cambridge, U.K., 2000.

[TC10] Quoc Trung Tran and Chee-Yong Chan. How to conquer why-not questions. In *SIGMOD*, pages 15–26, 2010.

CALL FOR RESEARCH PAPERS
ACM SIGMOD International Conference on
# MANAGEMENT OF DATA (SIGMOD 2011)
*June 12 - June 16, 2011, Athens, Greece*

http://www.sigmod2011.org

**General Chair:**
Timos Sellis (Inst. for the Mgmt. of Info. Systems)

**Program Chair:**
Renée J. Miller (University of Toronto)

**Proceedings Chairs:**
Anastasios Kementsietsidis (IBM Research – Thomas J. Watson Research Ctr.)
Yannis Velegrakis (University of Trento)

**Tutorial Chairs:**
Susan Davidson (University of Pennsylvania)
Tova Milo (Tel Aviv University)

**Keynote Chair:**
Pat Selinger (IBM Research - Almaden)

**Industrial Program Chair:**
Richard Hull (IBM Research – Thomas J. Watson Research Ctr.)

**Advisory Committee:**
Laura Hass (IBM Research - Almaden)
Donald Kossmann (ETH Zürich)

**Demo Chair:**
Chris Jermaine (Rice University)

**New Research Symposium Chairs:**
Yanlei Diao (University of Massachusetts)
Chris Olston (Yahoo! Research)

**Workshop Chair:**
Christian S. Jensen (Aarhus University)

**Undergraduate Research Program Chair:**
Irini Fundulaki (Institute of Computer Science – FORTH)

**Finance Chair:**
Mike Hatzopoulos (University of Athens)

**Publicity Chairs:**
Dieter Pfoser (Inst. for the Mgmt. of Info. Systems)
Ariel Fuxman (Microsoft Research)

**Sponsorship Chairs:**
Mike Carey (University of California – Irvine)
Vasilis Vassalos (Athens University of Economics & Business)

**Exhibits Chairs:**
Theodore Dalamagas (Inst. for the Mgmt. of Info. Systems)
Flip Korn (AT&T)

**Local Arrangements Chair:**
Yannis Kotidis (Athens U. of Economics & Business)

**Registration Chair:**
Alex Delis (University of Athens)

**Demonstration Local Arrangements Chair:**
Maria Halkidi (University of Piraeus)

**Web/Information Chair:**
Yannis Stavrakas (Inst. for the Mgmt. of Info. Systems)

The annual ACM SIGMOD conference is a leading international forum for database researchers, practitioners, developers, and users to explore cutting-edge ideas and results, and to exchange techniques, tools, and experiences. We invite the submission of original research contributions. We encourage submissions relating to all aspects of data management defined broadly, and particularly encourage work on topics of emerging interest in the research and development communities.

## TOPICS OF INTEREST

General areas of interests include, but are not limited to the following topics.

- New data management architectures, distributed data management (e.g., data stream management, P2P, replication, and cloud)
- Data management applications (e.g., Web services and mashups, social networks scientific databases, sensor networks, decision-making and analytics)
- Data models and languages (e.g., semi-structured, probabilistic, multi-media, temporal, spatial)
- Data-centric workflow and process management
- Data integration, meta-data management, data quality, data cleaning
- Performance, scalability and dependability of information systems
- Benchmarking and experimental methodology
- Other aspects of modern information systems such as data security, data privacy, personalization, user interfaces, etc.

## SUBMISSION GUIDELINES

All aspects of the submission and notification process will be handled electronically. Submissions must adhere to the paper formatting instructions.
Double-blind reviewing: As has become the tradition for SIGMOD, research papers will be judged for quality and relevance through double-blind reviewing, where the identities of the authors are withheld from the reviewers. Thus, author names and affiliation must not appear in the paper, and bibliographic references must be adjusted to preserve author anonymity.

Submissions should be uploaded through the submission site at:
https://cmt.research.microsoft.com/SIGMOD2011

## IMPORTANT DATES

Monday October 25, 2010: Abstract Submission, 3pm EDT
Monday November 1, 2010: Research Paper submission, 3pm EDT
Monday February 14, 2011: Notification of acceptance
Monday March 14, 2011: Final camera-ready papers due, 9am EDT

**PRELIMINARY CALL FOR PAPERS**

30th ACM SIGMOD–SIGACT–SIGART Symposium on

# PRINCIPLES OF DATABASE SYSTEMS (PODS 2011)

*June 13–June 15, 2011, Athens, Greece*


**Program Chair:**

Thomas Schwentick
Chair Computer Science I
TU Dortmund University
D-44221 Dortmund
Germany
thomas.schwentick@udo.edu



**Program Committee:**

Sara Cohen
*(Hebrew University of Jerusalem)*
Alin Deutsch
*(University of California, San Diego)*
Thomas Eiter
*(Vienna University of Technology)*
Wenfei Fan *(University of Edinburgh)*
Sudipto Guha
*(University of Pennsylvania)*
Claudio Gutierez *(University of Chile)*
Peter Haas
*(IBM Almaden Research Center)*
Ravi Kumar
*(Yahoo Research, Silicon Valley)*
Domenico Lembo
*(University of Rome La Sapienza)*
Thomas Lukasiewicz *(Oxford University)*
Filip Murlak *(Warsaw University)*
Dan Olteanu *(Oxford University)*
Rafail Ostrovsky
*(University of California Los Angeles)*
Thomas Schwentick
*(Chair, TU Dortmund University)*
Cristina Sirangelo *(ENS Cachan)*
Stijn Vansummeren
*(Université Libre de Bruxelles)*
Gerhard Weikum *(MPI Saarbrücken)*
Peter Widmayer *(ETH Zürich)*
Ke Yi
*(Hong Kong Univ. of Sci. and Tech.)*

**PODS General Chair:**

Maurizio Lenzerini
University of Rome La Sapienza

**Proceedings & Publicity Chair:**

Wim Martens
TU Dortmund University


The PODS symposium series, held in conjunction with the SIGMOD conference series, provides a premier annual forum for the communication of new advances in the theoretical foundations of database systems (see http://www.sigmod.org/the-pods-pages). For the 30th edition, original research papers providing new insights in the specification, design, or implementation of data-management tools are called for. We especially welcome papers addressing such insights in emerging database environments and applications. Topics that fit the interests of the symposium include the following (as they pertain to databases):

*languages for semi-structured data (including XML and RDF);*
*search query languages (including techniques from information retrieval);*
*distributed and parallel aspects of databases (including cloud computing);*
*dynamic aspects of databases (updates, views, real-time and sensor data, approximate query answering, data streams);*
*incompleteness, inconsistency, and uncertainty in databases;*
*schema and query extraction;*
*data integration; data exchange;*
*provenance; workflows;*
*metadata management; meta-querying;*
*semantic-Web data and ontologies;*
*data mining and machine learning techniques for databases;*
*constraints (specification, reasoning, mining, constraint databases);*
*privacy and security;*
*Web services; automatic verification of database-driven systems;*
*model theory, logics, algebras and computational complexity;*
*data modeling; data structures and algorithms for data management;*
*design, semantics, and optimization of query and database languages;*
*domain-specific databases (multi-media, scientific, spatial, temporal, text).*

Submitted papers should be at most twelve pages, including bibliography, using reasonable page layout and font size of at least 9pt (note that the SIGMOD style file does not have to be followed). Additional details may be included in an appendix, which, however, will be read at the discretion of the PC. *Papers longer than twelve pages (excluding the appendix) or in font size smaller than 9pt risk rejection without consideration of their merits.*

The submission process will be through the Web; a link to the submission website will appear on the conference website in due time. Note that, unlike the SIGMOD conference, PODS does not use double-blind reviewing, and therefore PODS submissions should be eponymous (i.e., the names and affiliations of authors should be listed on the paper).

The results must be unpublished and not submitted elsewhere, including the formal proceedings of other symposia or workshops. Authors of an accepted paper will be expected to sign copyright release forms, and one author is expected to present it at the conference.

**Important Dates:**

| | | | |
|---|---|---|---|
| Short abstracts due: | 7 | December | 2010 |
| Paper submission: | 14 | December | 2010 |
| Notification: | 7 | March | 2011 |
| Camera-ready copy: | 4 | April | 2011 |

**Best Paper Award:** An award will be given to the best submission, as judged by the PC.

**Best Student Paper Award:** There will also be an award for the best submission, as judged by the PC, written exclusively by a student or students. An author is considered as a student if at the time of submission, the author is enrolled in a program at a university or institution leading to a doctoral/master's/bachelor's degree.

The PC reserves the right to give both awards to the same paper, not to give an award, or to split an award among several papers. Papers authored or co-authored by PC members are not eligible for an award.

# Call for Participation

The **27th IEEE International Conference on Data Engineering** 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. The mission of the conference is to share research solutions to problems of today's information society and to identify new issues and directions for future research and development work.

**Keynotes**

- Anastasia Ailamaki (EPFL, Switzerland)
- Johannes Gehrke (Cornell University, USA)
- Georg Gottlob (Oxford University, UK)

**Workshops**

- 6th International Workshop on Self Managing Database Systems (SMDB 2011)
- 1st International Workshop on Managing Data Throughout its Lifecycle (DaLi 2011)
- 2nd International Workshop on Graph Data Management: Techniques and Applications (GDM 2011)
- 3rd Workshop on Hot Topics in Software Upgrades (HotSWUp 2011)
- Data Engineering Applications in Emerging Areas - Health Care and Energy Informatics
- 2nd International Workshop on Data Engineering meets the Semantic Web (DESWeb 2011)
- Analytics and Data Management for Semi-Structured Business Processes
- ICDE PhD Workshop

**Venue**

ICDE 2011 will be held in Hannover, Germany. Hannover is the capital of the federal state of Lower Saxony (Niedersachsen), Germany. Hannover is known as a trade fair city (e.g. CeBIT) but also for their Royal Gardens of Herrenhausen and many other places of interest.

**For more information please regular visit the conference web site:**

**http://www.icde2011.org/**

IEEE computer society

IEEE Technical Committee on Data Engineering

Microsoft

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

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