Datalog and Emerging Applications: An Interactive Tutorial

Shan Shan Huang
LogicBlox, Inc.
Atlanta, GA USA
ssh@logicblox.com

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

Boon Thau Loo
Dept. of CIS
University of Pennsylvania
Philadelphia, PA USA
boonloo@cis.upenn.edu

ABSTRACT

We are witnessing an exciting revival of interest in recursive Datalog queries in a variety of emerging application domains such as data integration, information extraction, networking, program analysis, security, and cloud computing. This tutorial briefly reviews the Datalog language and recursive query processing and optimization techniques, then discusses applications of Datalog in three application domains: data integration, declarative networking, and program analysis. Throughout the tutorial, we use LogicBlox, a commercial Datalog engine for enterprise software systems, to allow the audience to walk through code examples presented in the tutorial.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Query languages

General Terms

Languages

Keywords

Datalog, recursive query processing, data integration, declarative networking, program analysis

1. INTRODUCTION

Mainstream interest in Datalog in the database systems community flourished in the eighties and early nineties, but a perceived lack of compelling applications at the time [39] ultimately forced Datalog research into a long dormancy. In recent years, however, Datalog has suddenly re-emerged at the center of a wide range of new applications, including data integration [26, 17, 22], declarative networking [29, 28, 27], program analysis [13], information extraction [19, 38], network monitoring [5], security [31, 25], and cloud computing [7]. A common thread across these systems is the use of the Datalog language as a higher level abstraction for querying graphs and relational structures, efficient recursive query execution and incremental view maintenance techniques based on the relational model, and formal reasoning and analysis. Each application domain takes the core Datalog language and then further customizes and extends the core language and implementation techniques to meet its particular needs.

As the list of applications above indicates, interest today in Datalog extends well beyond the core database community. Indeed, the successful *Datalog 2.0 Workshop* held in March 2010 at Oxford University attracted over 100 attendees from a wide range of areas (including databases, programming languages, verification, security, and AI). Moreover, there has also been a surprising resurgence in *commercial* interest in Datalog, led by startups such as Lixto [19] (information extraction), LogicBlox [3] (enterprise decision automation) and Semmle [4] (program analysis).

We feel that the time is right for a "re-introduction" of the mainstream database systems community to Datalog, as seen through the lens of these recent developments. The tutorial first briefly reviews the Datalog language and recursive query processing and optimization techniques, covering the basics but emphasizing features and techniques beyond "classical" Datalog which are vital for practical applications. The tutorial will focus on discussing three applications of Datalog in emerging domains from the list above: data integration and exchange, declarative networking, and program analysis. We also include a case study of the LogicBlox system.

Compared to prior surveys [10, 14, 36] and textbook presentations [6, 12, 40], we can present our material from the perspective of modern, practical applications and commercial systems, several of which have significantly more mature and complete Datalog implementations (including optimizations) compared to the state-of-the-art of a decade ago. This tutorial accompanies our own survey paper of Datalog from a modern perspective [21].

A unique feature of the tutorial is its *interactive* nature: we plan to distribute software and academic licenses of the LogicBlox system, along with Datalog programs corresponding to the examples used throughout the tutorial.

2. REVIEW OF FOUNDATIONS

The tutorial will begin with a brief review of the foundations of Datalog, with a focus on language and semantics, query processing, and optimizations. Given the scope of the topics covered, we will necessarily assume prior familiarity with the basics of Datalog, and will cover the foundational material at a brisk pace. In addition, we limit the scope of the presentation by focusing on key aspects that are useful in today's practical systems.

Throughout, the presentation will include running examples that can be executed interactively by participants using

SIGMOD'11, June 12–16, 2011, Athens, Greece.
ACM 978-1-4503-0661-4/11/06.

LogicBlox. We briefly touch upon language extensions, but defer the bulk of the discussion to the second half of the tutorial, when we discuss extensions in the context of their motivating applications.

3. APPLICATIONS AND SYSTEMS

The bulk of the tutorial will focus on emerging Datalog-based applications and systems. We present three representative application domains: data integration, declarative networking, and program analysis. In each domain, we highlight language extensions, runtime considerations, and use cases. We also survey advanced features of academic and commercial systems.

3.1 Data Integration

Broadly speaking, the goal of data integration [26] is to provide mechanisms for tying together collections of databases with heterogeneous schemas, semantic mismatches, and varying capabilities so that they can be accessed and used as an integrated unit. In the classical data integration scenario, we are given a *source* database schema (or schemas), a *target* schema, a collection of *schema mappings* relating source and target schemas, and a source database instance (or instances). In *virtual data integration* [18], we are also given a query over the target schema. The goal is to *reformulate* the query so that it can be answered over the source databases. In *data exchange* [17], the goal is to compute a target database instance so that queries over the target schema can be answered directly.

Schema mappings are typically specified using *tuple generating dependencies* [11]. These are logical constraints that correspond very naturally to Datalog rules, provided we enrich Datalog with *Skolem terms* to play the role of existentially-quantified variables in schema mappings. Both virtual and materialized data integration then become just special cases of Datalog query evaluation. However, a significant issue in this extension is that termination is no longer guaranteed. A variety of syntactic restrictions guaranteeing termination in PTIME have been proposed. We present one such notion called *weak acyclicity* [16, 17].

3.2 Declarative Networking

Declarative networking is based on the observation that recursive queries are a natural fit for expressing network protocols, which themselves are based on recursive relations among nodes in the network. Intuitively, one can view the forwarding tables generated by network protocols as the output of distributed recursive queries over changing input network state (network links, nodes, operator policies, etc.), and the query results need to be kept consistent with the changing network state.

Using a few representative routing protocol examples, we will describe the *Network Datalog* [27] language used in declarative networking. In comparison to traditional Datalog, Network Datalog is enhanced to capture typical network realities including distribution, link-layer constraints on communication (and hence deduction), and soft-state semantics.

We also present the *pipelined semi-naïve evaluation* [27] strategy, which relaxes semi-naïve's requirement of synchronized "lock-step" iterations, and enables query evaluation to work much more efficiently in an asynchronous distributed setting. Finally, we introduce a provenance-based approach [32] for incremental recursive maintenance that needs far less bandwidth than DRed in a distributed setting.

3.3 Program Analysis

Program analysis covers a broad range of analyses, from data-flow and control-flow, to pointer analysis, to static code structure. The results of these analyses are used for optimization, bug discovery, enforcement of coding standards, etc. Program analyses are highly recursive in nature, making Datalog a natural fit. For instance, in points-to analysis, the heap locations pointed to by a variable are the union of its direct heap allocations assignments with the locations pointed to by its aliases.

We discuss two recent tools, *.QL* [23] and *Doop* [13]. We show that Datalog can be effectively used to describe at a high level otherwise complex algorithms. We also show that non-trivial (e.g. non-linear) recursion is necessary in program analysis. Furthermore, we show that a Datalog implementation of pointer analysis (as in Doop) can outperform highly hand-tuned implementations in lower level imperative languages like Java.

3.4 Commercial and Academic Systems

We discuss two active commercial systems, LogicBlox [3] and Semmle [4], and conclude with a (partial) overview of (past and present) academic projects. We highlight their distinguishing features.

LogicBlox and Semmle each use novel, non-trivial type systems to catch likely programming errors and aid query optimization [15]. Both systems support module mechanisms. LogicBlox additionally supports meta-programming, a reusability feature that has proven useful both within the company, and in areas such as security and distributed query processing [31]. LogicBlox also supports an update language, Skolem terms, user-defined functions, and integrity constraints.

We also review two important academic systems from the "classical age" of Datalog research, *Coral* [35] and *LDL++* [9]. Each supports advanced forms of recursion through negation. Furthermore, LDL++ supports a syntax for user-defined aggregations such that, if they can be shown to be monotone, then their use in recursion is unrestricted. In addition to a module mechanism, LDL++ also implements a limited form of meta-programming. Finally, we highlight the ongoing *BOOM* project, based on a Datalog dialect called *Dedalus* [8]. Designed for distributed programming, a major novelty of Dedalus is its explicit representation of *time*.

4. OPEN ISSUES

We close the tutorial with a brief sampling of open challenges.

Datalog evaluation on emerging architectures. The end of Moore's Law has stimulated the emergence of new computing architectures. GPUs and FPGAs are readily available in commodity hardware [2], and even on the cloud as services [1]. Techniques have been proposed for query processing using GPUs [20, 24] and FPGAs [34, 33]. However, little has been done to address the challenges of evaluating recursive queries on these new architectures.

Datalog as a general purpose programming language. Datalog has traditionally been viewed as a query processing language. Recently, however, Datalog has been used for general purpose computing, such as describing security protocols [31], or building the entire enterprise application stack. These new applications place interesting demands on both the expressiveness and performance of Datalog. Efficient implementations of advanced forms of recursion through negation [37], and explicit representations of time [30], remain to be demonstrated.

Extensions, safety, and complexity. Related to the last point, practical applications invariably require extensions to the core language, such as arithmetic or Skolem functions, but these extensions destroy Datalog's attractive guarantees of PTIME termination. This phenomenon has been examined in a number of papers, but our understanding remains far from complete. We point out some gaps in the literature and argue for a renewed push to close them.

Biographies

Shan Shan Huang is the lead of compiler development at LogicBlox, Inc. Her work focuses on enriching Datalog with features necessary for building large enterprise applications, while keeping an vigilant eye on maintaining proper semantics and efficient evaluation.

Todd J. Green is an Assistant Professor of Computer Science at the University of California, Davis. His research interests include data integration, data provenance, incomplete and probabilistic databases, query optimization, semistructured data, and streaming data processing.

Boon Thau Loo is an Assistant Professor in the Computer and Information Science department at the University of Pennsylvania. His research focuses on distributed data management systems, Internet-scale query processing, and the application of data-centric techniques and formal methods to the design, analysis and implementation of networked systems.

5. REFERENCES

[1] Amazon High Performance Computing Clusters. `http://aws.amazon.com/ec2/hpc-applications`. Accessed Dec 2010.

[2] Intel Atom E600C with FPGA. `http://www.intel.com/design/intarch/atom/index.htm`. Accessed Dec 2010.

[3] LogicBlox. `http://www.logicblox.com`. Accessed Dec 2010.

[4] Semmle. `http://www.semmle.com`. Accessed Dec 2010.

[5] Abiteboul, S., Abrams, Z., Haar, S., and Milo, T. Diagnosis of Asynchronous Discrete Event Systems—Datalog to the Rescue! In *PODS* (2005).

[6] Abiteboul, S., Hull, R., and Vianu, V. *Foundations of Databases*. Addison-Wesley, 1995.

[7] Alvaro, P., Condie, T., Conway, N., Elmeleegy, K., Hellerstein, J. M., and Sears, R. Boom analytics: exploring data-centric, declarative programming for the cloud. In *EuroSys* (2010).

[8] Alvaro, P., Marczak, W., Conway, N., Hellerstein, J. M., Maier, D., and Sears, R. C. Dedalus: Datalog in time and space. Tech. Rep. UCB/EECS-2009-173, EECS Department, University of California, Berkeley, Dec 2009.

[9] Arni, F., Ong, K., Tsur, S., Wang, H., and Zaniolo, C. The deductive database system LDL++. *TPLP 3*, 1 (2003), 61–94.

[10] Bancilhon, F., and Ramakrishnan, R. An amateur's introduction to recursive query processing strategies. *SIGMOD Rec. 15*, 2 (1986), 16–52.

[11] Beeri, C., and Vardi, M. Y. A proof procedure for data dependencies. *J. ACM 31*, 4 (1984), 718–741.

[12] Bidoit, N. *Bases de Données Déductives: Présentation de Datalog*. Armand Colin, 1992.

[13] Bravenboer, M., and Smaragdakis, Y. Strictly declarative specification of sophisticated points-to analyses. In *OOPSLA* (2009).

[14] Ceri, S., Gottlob, G., and Tanca, L. What you always wanted to know about datalog (and never dared to ask). *TKDE 1*, 1 (1989), 146–166.

[15] de Moor, O., Sereni, D., Avgustinov, P., and Verbaere, M. Type inference for datalog and its application to query optimisation. In *PODS* (2008).

[16] Deutsch, A., and Tannen, V. Reformulation of xml queries and constraints. In *ICDT* (2003), pp. 225–241.

[17] Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. Data exchange: semantics and query answering. *TCS 336*, 1 (2005), 89–124.

[18] Genesereth, M. R. *Data Integration: The Relational Logic Approach*. Morgan & Claypool Publishers, 2010.

[19] Gottlob, G., Koch, C., Baumgartner, R., Herzog, M., and Flesca, S. The Lixto data extraction project: back and forth between theory and practice. In *PODS* (2004).

[20] Govindaraju, N., Gray, J., Kumar, R., and Manocha, D. GPUTeraSort: high performance graphics co-processor sorting for large database management. In *SIGMOD* (2006).

[21] Green, T. J., Huang, S. S., and Loo, B. T. Datalog and recursive query processing. *Foundations and Trends in Databases* (2011). In preparation.

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

[23] Hajiyev, E., Verbaere, M., and de Moor, O. Codequest: Scalable source code queries with datalog. In *ECOOP* (2006).

[24] He, B., Lu, M., Yang, K., Fang, R., Govindaraju, N. K., Luo, Q., and Sander, P. V. Relational query coprocessing on graphics processors. *ACM TODS 34* (December 2009).

[25] Jim, T. SD3: A Trust Management System With Certified Evaluation. In *IEEE Symposium on Security and Privacy* (May 2001).

[26] Lenzerini, M. Data integration: A theoretical perspective. In *PODS* (2002), pp. 233–246.

[27] Loo, B. T., Condie, T., Garofalakis, M., Gay, D. E., Hellerstein, J. M., Maniatis, P., Ramakrishnan, R., Roscoe, T., and Stoica, I. Declarative networking: Language, execution and optimization. In *SIGMOD* (2006).

[28] Loo, B. T., Condie, T., Hellerstein, J. M., Maniatis, P., Roscoe, T., and Stoica, I. Implementing declarative overlays. In *SOSP* (2005).

[29] Loo, B. T., Hellerstein, J. M., Stoica, I., and Ramakrishnan, R. Declarative routing: extensible routing with declarative queries. In *SIGCOMM* (2005).

[30] Ludäscher, B., May, W., and Lausen, G. Nested transactions in a logical language for active rules. In *LID* (1996).

[31] Marczak, W. R., Huang, S. S., Bravenboer, M., Sherr, M., Loo, B. T., and Aref, M. Secureblox: customizable secure distributed data processing. In *SIGMOD* (2010).

[32] Mengmeng Liu and Nicholas Taylor and Wenchao Zhou and Zachary Ives and Boon Thau Loo. Recursive Computation of Regions and Connectivity in Networks. In *ICDE* (2009).

[33] Mueller, R., Teubner, J., and Alonso, G. Data processing on FPGAs. *PVLDB 2*, 1 (August 2009).

[34] Mueller, R., Teubner, J., and Alonso, G. Glacier: a query-to-hardware compiler. In *SIGMOD* (2010).

[35] Ramakrishnan, R., Srivastava, D., Sudarshan, S., and Seshadri, P. The CORAL deductive system. *The VLDB Journal 3*, 2 (1994), 161–210.

[36] Ramakrishnan, R., and Ullman, J. D. A Survey of Research on Deductive Database Systems. *JLP 23*, 2 (1993).

[37] Ross, K. A syntactic stratification condition using constraints. In *ILPS* (1994).

[38] Shen, W., Doan, A., Naughton, J., and Ramakrishnan, R. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB* (2007).

[39] Stonebraker, M., and Hellerstein, J. M., Eds. *Readings in Database Systems, Third Edition.* Morgan Kaufmann, 1998.

[40] Ullman, J. D. *Principles of Database and Knowledge-Base Systems*, vol. II. W. H. Freeman & Co., 1990.