Datalog and Recursive Query Processing

Todd J. Green
LogicBlox Inc.
todd.green@logicblox.com

Shan Shan Huang
LogicBlox Inc.
ssh@logicblox.com

Boon Thau Loo
University of Pennsylvania
boonloo@cis.upenn.edu

Wenchao Zhou
Georgetown University
wzhou@cs.georgetown.edu

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

T. J. Green, S. Huang, B. T. Loo, W. Zhou. *Datalog and Recursive Query Processing*. Foundations and Trends® in Databases, vol. 5, no. 2, pp. 105–195, 2012.

This Foundations and Trends® issue was typeset in LATEX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-60198-753-2
© 2013 T. J. Green, S. Huang, B. T. Loo, W. Zhou

Foundations and Trends® in Databases
Volume 5, Issue 2, 2012

Editorial Scope

Topics

Foundations and Trends® in Databases covers a breadth of topics relating to the management of large volumes of data. The journal targets the full scope of issues in data management, from theoretical foundations, to languages and modeling, to algorithms, system architecture, and applications. The list of topics below illustrates some of the intended coverage, though it is by no means exhaustive:

- Data models and query languages
- Query processing and optimization
- Storage, access methods, and indexing
- Transaction management, concurrency control, and recovery
- Deductive databases
- Parallel and distributed database systems
- Database design and tuning
- Metadata management
- Object management
- Trigger processing and active databases
- Data mining and OLAP
- Approximate and interactive query processing
- Data warehousing
- Adaptive query processing
- Data stream management
- Search and query integration
- XML and semi-structured data
- Web services and middleware
- Data integration and exchange
- Private and secure data management
- Peer-to-peer, sensornet, and mobile data management
- Scientific and spatial data management
- Data brokering and publish/subscribe
- Data cleaning and information extraction
- Probabilistic data management

Information for Librarians

Foundations and Trends® in Databases, 2012, Volume 5, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Foundations and Trends® in Databases
Vol. 5, No. 2 (2012) 105–195
© 2013 T. J. Green, S. Huang, B. T. Loo, W. Zhou
DOI: 10.1561/1900000017

Datalog and Recursive Query Processing

Todd J. Green
LogicBlox Inc.
todd.green@logicblox.com

Shan Shan Huang
LogicBlox Inc.
ssh@logicblox.com

Boon Thau Loo
University of Pennsylvania
boonloo@cis.upenn.edu

Wenchao Zhou
Georgetown University
wzhou@cs.georgetown.edu

Contents

Abstract

In recent years, we have witnessed a revival of the use of recursive queries in a variety of emerging application domains such as data integration and exchange, information extraction, networking, and program analysis. A popular language used for expressing these queries is Datalog. This paper surveys for a general audience the Datalog language, recursive query processing, and optimization techniques. This survey differs from prior surveys written in the eighties and nineties in its comprehensiveness of topics, its coverage of recent developments and applications, and its emphasis on features and techniques beyond "classical" Datalog which are vital for practical applications. Specifically, the topics covered include the core Datalog language and various extensions, semantics, query optimizations, magic-sets optimizations, incremental view maintenance, aggregates, negation, and types. We conclude the paper with a survey of recent systems and applications that use Datalog and recursive queries.

T. J. Green, S. Huang, B. T. Loo, W. Zhou. *Datalog and Recursive Query Processing*. Foundations and Trends® in Databases, vol. 5, no. 2, pp. 105–195, 2012.

DOI: 10.1561/1900000017.

1 Introduction

Mainstream interest in Datalog in the database systems community flourished in the eighties and early nineties. During this period, there were several pioneering Datalog systems, primarily from academia. Two of the more prominent ones with complete implementations include Coral [99] and LDL++ [20]. Some ideas from these early research prototypes made it into mainstream commercial databases. For instance, Oracle, DB2, and SQL Server provide support for limited forms of support for recursion, based on SQL-99 standards. However, a perceived lack of compelling applications at the time [113] ultimately forced Datalog research into a long dormancy, and stifled its use in practice. Coral and LDL++ ceased active development in 1997 and 2000 respectively, and commercial systems did not extend a limited form of Datalog.

In recent years, however, Datalog has reemerged at the center of a wide range of new applications, including data integration [68, 43, 50], declarative networking [80, 77, 75], program analysis [29], information extraction [110], network monitoring [10], security [85, 60], optimizations [73], and cloud computing [15, 16]. Compared to the state-of-the-art of two decades ago, the modern systems that drives these emerging applications have significantly more mature and complete Datalog implementations, and often times deploy applications that are orders of magnitude larger in code size and complexity compared to the older generation of Datalog programs.

In terms of modern academic systems, the IRIS reasoner [59] is an open-source general purpose Datalog execution engine with support for optimizations, stratified and locally stratified negation. There are also publicly available Datalog systems tailored for specific applications. These include the Orchestra system for collaborative data sharing [92], BDDBDD [24] for program analysis, the RapidNet [101] declarative networking platforms, and the Bloom [16] platform for declarative programming in the cloud.

In the commercial world, a major development is the emergence of enterprise Datalog systems, most notably LogicBlox [4], Datomic [2], Semmle [7], and Lixto [49]. Semmle and Lixto are targeted at specific domains of program analysis and information extraction respectively, while LogicBlox and Datomic aim to provide a general platform for developing enterprise software.

The revival of Datalog in the new generation of applications is driven by the increasing need for high-level abstractions for reasoning about and rapidly developing complex systems that process large amounts of data, and are sometimes distributed and parallel. Datalog provides a declarative interface that allows the programmer to focus on the tasks (“what”), not the low-level details (“how”). A common thread across these systems is the use of the Datalog language as a declarative abstraction for querying graphs and relational structures, and implementing iterations and recursions. Its clear and simple syntax with well understood semantics aims to achieve the best of both worlds – having a rich enough language to support a wide range of applications, yet at a high and concise level that makes rapid prototyping easy for programmers without having to worry about low level messy details related to robustness and parallelism. The high-level specifications also make code analysis easier, for applying optimizations and for reasoning about transactions and safety.

1.1 Contributions and Roadmap

This survey paper aims to provide an accessible and gentle introduction to Datalog and recursive query processing to readers with some basic background in databases (in particular, SQL and the relational model). Given the wide range of research literature on Datalog spanning decades, we identify a "practical" subset of Datalog based on recent advances in the adoption of Datalog. In particular, our survey aims to cover the following:

- **Language.** Core Datalog syntax and semantics. (Chapter 2)
- **Query processing.** Recursive query processing techniques for executing Datalog programs efficiently, using the bottom-up and top-down evaluation strategies, such as the well-known *semi-naïve* [22, 21] and *Query/Subquery (QSQ)* [67] evaluation strategies. (Chapter 3)
- **Incremental maintenance.** Extensions to query processing techniques in the previous chapter, to include mechanisms for incrementally updating the materialized views of a Datalog program, as the input data changes, without having to recompute the entire Datalog program from scratch. (Chapter 4)
- **Common extensions.** 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. Here, we discuss extensions to incorporate negation, aggregation, arithmetic, uninterpreted functions, and updates, as well as the query processing techniques to handle these extensions. (Chapter 5).

The survey concludes in Chapter 6 with a brief survey of recent applications of Datalog, in the domains of program analysis, declarative networking, data integration and exchange, enterprise software systems, etc.

1.2 Relationship with Previous Surveys

Our survey serves as an entry point into several other survey papers and books on Datalog. We briefly mention some of them:

- Bancilhon et al. [23] surveys and compares various strategies for processing and optimizing recursive queries in a greater depth compared to our survey.
- Ceri et al. [32] presents the syntax and semantics of Datalog along with evaluation and optimization techniques for efficient execution. Extensions to the Datalog language, such as built-in predicates and negation are also discussed.
- Ramakrishnan and Ullman [100] provides a high-level overview of the Datalog language, query evaluation and optimizations, and more advanced topics on negation and aggregation in a few pages. This should be viewed as a "quick-starter" guide for someone exposed to Datalog for the first time.
- Textbooks [12, 27, 33, 118, 36] cover some topics (e.g. language, semantics, magic sets) in greater detail than our survey. Abiteboul et al. [12] in particular is a widely used textbook geared towards a database theory audience.

Overall, our survey is broader than Bancilhon [23], which focuses primarily on query processing, and Ramakrishnan and Ullman [100], which surveys Datalog systems (which are now more than a decade old) with a brief discussion on query processing and optimizations. We cover a breath of topics similar to the surveys [32, 88], but provide significantly more details on systems issues related to query processing, incremental maintenance, and modern applications. Compared to all of the above surveys, we provide a more systems approach in presentation of classical topics, and discuss only extensions relevant to modern applications.

1.3 First Example: All-Pairs Reachability

We begin with a high level introduction to the Datalog language and its basic evaluation strategy. As our first example, we consider a Datalog program that computes *all-pairs reachability*, essentially a transitive closure computation in a graph for figuring out all pairs of nodes that are connected (reachable) to each other.

```
r1 reachable(X,Y) :- link(X,Y).
r2 reachable(X,Y) :- link(X,Z), reachable(Z,Y).
query(X,Y) :- reachable(X,Y).
```

The above two rules, named as *r1* and r2, derive the `reachable` nodes (i.e. `reachable(X,Y)` using facts about directly linked nodes (i.e. `link(X,Y)`). Here, we use capital letters `X` and `Y` to signify that they are variables in the domain of all the nodes. The output of interest in this program, as denoted by the special predicate `query(X,Y)`, is the set of derived `reachable` facts. The input graph in this case can represent a network of routers, and forms a basis for implementing network routing protocols [80], web crawlers [81], and network crawlers [79].

Rule *r1* expresses that node `X` is reachable from `Y` (i.e. `reachable(X,Y)`) if they are directly linked. Rule *r2* is a bit more interesting, as it specifies the `reachable` relation in terms of itself: `(X,Y)` are reachable from one another if `X` has a direct link to a node (`Z`) that is reachable to `Y`. We refer to rules such as *r2* as *recursive* rules, since the *reachable* relation appears in both the rule body (right of " `:-` ") and head (left of " `:-` "). Rule *r2* is also a *linear* recursive rule ,since `reachable` appears only once in the rule body.

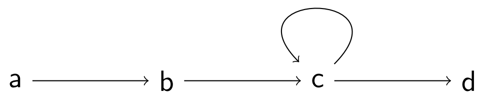

Figure 1.1: Example graph used for reachability computation.

We illustrate the execution of Datalog rules by evaluating the `reachable` rules over the graph shown in Figure 1.1, which depicts a network consisting of three nodes and four direct `links`. Thus, there are

(initial base tuples)

`link`

X	Y
a	b
b	c
c	c
c	d

(iteration 1)

`reachable`

X	Y
a	b
b	c
c	c
c	d

(iteration 2)

`reachable`

X	Y
a	b
b	c
c	c
c	d
a	c
b	c
b	d
c	d

(iteration 3)

`reachable`

X	Y
a	b
b	c
c	c
c	d
a	c
b	d
a	d

Figure 1.2: Tuples derived by the *All-pairs Reachability* program for each iteration. New tuples derived in the current iteration that are not known in prior iterations are shaded.

four initial entries (tuples) in `link`: `link(a,b)`, `link(b,c)`, `link(c,c)`, and `link(c,d)`.

Intuitively, rule evaluation can be understood as the repeated application of rules over existing tuples to derive new tuples, until no more new tuples can be derived (i.e. evaluation has reached a *fixpoint*). Each application of rules over existing tuples is referred to as an *iteration*. This evaluation strategy is often times referred to as the *naïve* evaluation strategy.

The evaluation of the reachability rules over the network in Figure 1.1 reaches a fixpoint in three iterations, as shown in Figure 1.2. In iteration 1, rule *r1* takes as input the initial `link` tuples, and use that to generate 4 `reachable` tuples. These tuples essentially represent all pairs of nodes reachable within one hop. In the next two iterations, all `reachable` tuples generated in previous iterations are used as input to rule *r2* to generate more `reachable` tuples that are two and three hops apart. Iteration 4 (not shown in the figure) derives the same set of tuples as iteration 3, and hence, a fixpoint is reached. Given that no two nodes are separated by more than 3 hops, the recursive query completes in 4 iterations.

As an optimization, instead of using all derived facts as input to rules at each iteration, one can suppress the evaluation that uses *only* tuples already learned in prior iterations when computing new tuples the next iteration. For instance, when generating new facts in iteration 3, rule *r2* will not evaluate for inputs `reachable(b,c)` and `link(a,b)`, since they have already been used in iteration 1. The intuitive description above corresponds loosely to the *semi-naïve* evaluation strategy, which will be described in greater detail in Chapter 3.

Note that the above approach is a *bottom-up* evaluation technique, where existing facts are used as input to rule bodies to derive new facts. A fixpoint is reached when no new facts are derived. This is also known as a *forward-chaining* style of evaluation. An alternative approach used in Prolog [112] uses a goal-oriented *backward-chaining* approach, starting from the goal (i.e. query), and then expanding the rule bodies in a top-down fashion.

A top-down approach allows for an evaluation strategy that focuses only on facts necessary for the goal. However, a bottom-up evaluation approach used in Datalog allows us to draw upon a wealth of query processing and optimization techniques to draw upon for doing the computations efficiently even when datasets are too large to fit in main memory. Moreover, as we show in Section 3.3, query optimization techniques can optimize Datalog programs for bottom-up evaluation, to avoid deriving facts not relevant to answering queries.

References

[1] *BioPerl, http://bioperl.org.*

[2] *Datomic website, http://www.datomic.com/.*

[3] *H2 Database Engine, http://www.h2database.com.*

[4] *LogicBlox website, http://www.logicblox.com/.*

[5] *Microsoft SQL server, http://www.microsoft.com/sql.*

[6] *PostgreSQL, http://www.postgresql.org/.*

[7] *Semmle Web site, http://www.semmle.com.*

[8] *uBio, http://www.ubio.org.*

[9] S. Abiteboul, E. Simon, and V. Vianu. Non-deterministic languages to express deterministic transformations. In *PODS*, 1990.

[10] Serge Abiteboul, Zoe Abrams, Stefan Haar, and Tova Milo. Diagnosis of Asynchronous Discrete Event Systems—Datalog to the Rescue! In *PODS*, 2005.

[11] Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialized views. In *PODS*, 1998.

[12] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[13] Serge Abiteboul and Victor Vianu. Datalog extensions for database queries and updates. *J. Comput. Syst. Sci.*, 43:62–124, August 1991.

[14] Foto Afrati, Stavros S. Cosmadakis, and Mihalis Yannakakis. On datalog vs. polynomial time. In *PODS*, 1991.

[15] Peter Alvaro, Tyson Condie, Neil Conway, Khaled Elmeleegy, Joseph M. Hellerstein, and Russell Sears. Boom analytics: exploring data-centric, declarative programming for the cloud. In *EuroSys*, 2010.

[16] Peter Alvaro, Neil Conway, Joseph M. Hellerstein, and William R. Marczak. Consistency analysis in bloom: a calm and collected approach. In *CIDR*, 2011.

[17] Peter Alvaro, William Marczak, Neil Conway, Joseph M. Hellerstein, David Maier, and Russell C Sears. Dedalus: Datalog in time and space. Technical Report UCB/EECS-2009-173, EECS Department, University of California, Berkeley, Dec 2009.

[18] Tom J. Ameloot, Frank Neven, and Jan Van den Bussche. Relational Transducers for Declarative Networking. In *PODS*, 2011.

[19] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. pages 89–148, 1988.

[20] Faiz Arni, KayLiang Ong, Shalom Tsur, Haixun Wang, and Carlo Zaniolo. The deductive database system LDL++. *TPLP*, 3(1):61–94, 2003.

[21] I. Balbin and K. Ramamohanarao. A generalization of the differential approach to recursive query evaluation. *Journal of Logic Programming*, 4(3), 1987.

[22] Francois Bancilhon. Naive evaluation of recursively defined relations. *On Knowledge Base Management Systems: Integrating AI and DB Technologies*, 1986.

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

[24] BDD-Based Deductive DataBase. `http://bddbddb.sourceforge.net/`.

[25] Catriel. Beeri and Raghu. Ramakrishnan. On the power of magic. In *PODS*, 1987.

[26] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

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

[28] Martin Bravenboer and Yannis Smaragdakis. *Doop website, http://doop.program-analysis.org/.*

[29] Martin Bravenboer and Yannis Smaragdakis. Strictly declarative specification of sophisticated points-to analyses. In *OOPSLA*, 2009.

[30] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[31] Dario Campagna, Beata Sarna-Starosta, and Tom Schrijvers. Optimizing Inequality Joins in Datalog with Approximated Constraint Propagation. In Claudio Russo and Neng-Fa Zhou, editors, *Practical Aspects of Declarative Languages, 14th International Symposium, Proceedings*. Springer, 2012.

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

[33] Stefano Ceri, Georg Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.

[34] Keith L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322, 1977.

[35] Sara Cohen, Joseph Gil, and Evelina Zarivach. Datalog programs over infinite databases, revisited. In *DBPL*, 2007.

[36] Robert M. Colomb. *Deductive Databases and their Applications*. Taylor and Francis, 1998.

[37] Neil Conway, William R. Marczak, Peter Alvaro, Joseph M. Hellerstein, and David Maier. Logic and lattices for distributed programming. In *SoCC*, 2012.

[38] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, 1977.

[39] Steven Dawson, C. R. Ramakrishnan, and David S. Warren. Practical program analysis using general purpose logic programming systems—a case study. In *PLDI*, 1996.

[40] Alin Deutsch and Val Tannen. Reformulation of xml queries and constraints. In *ICDT*, pages 225–241, 2003.

[41] Guozhu Dong, Leonid Libkin, Jianwen Su, and Limsoon Wong. Maintaining transitive closure of graphs in sql. *In Int. J. Information Technology*, 5, 1999.

[42] Guozhu Dong and Jianwen Su. Incremental and decremental evaluation of transitive closure by first-order queries. *Inf. Comput.*, 120(1):101–106, 1995.

[43] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *TCS*, 336(1):89–124, 2005.

[44] John Field, Maria-Cristina Marinescu, and Christian Stefansen. Reactors: A data-oriented synchronous/asynchronous programming model for distributed applications. *Theor. Comput. Sci.*, 410(2-3):168–201, 2009.

[45] Jörg Flum, Max Kubierschky, and Bertram Ludäscher. Total and partial well-founded datalog coincide. In *ICDT*, 1997.

[46] Jörg Flum, Max Kubierschky, and Bertram Ludäscher. Games and total datalog$^\neg$ queries. *Theoretical Computer Science*, 239(2):257–276, 2000.

[47] Allen Van Gelder. The alternating fixpoint of logic programs with negation. *JCSS*, 47(1):185 – 221, 1993.

[48] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080, 1988.

[49] Georg Gottlob, Christoph Koch, Robert Baumgartner, Marcus Herzog, and Sergio Flesca. The Lixto data extraction project: back and forth between theory and practice. In *PODS*, 2004.

[50] Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen. Update exchange with mappings and provenance. In *VLDB*, 2007.

[51] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, 2007.

[52] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In *SIGMOD*, 1993.

[53] GUS: The Genomics Unified Schema. `http://www.gusdb.org/`.

[54] Elnar Hajiyev, Mathieu Verbaere, and Oege de Moor. Codequest: Scalable source code queries with datalog. In David Thomas, editor, *ECOOP*, 2006.

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

[56] Y. Halevy, G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB Journal*, 14(1):68–83, 2005.

[57] Joseph M. Hellerstein. Declarative imperative: Experiences and conjectures in distributed logic. 2010. SIGMOD Record 39(1).

[58] Neil Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1-3):86–104, 1986.

[59] IRIS (Integrated Rule Inference System) Reasoner. `http://www.iris-reasoner.org/`.

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

[61] David B. Kemp. Efficient recursive aggregation and negation in deductive databases. *TKDE*, 10(5), 1998.

[62] Michael Kifer. On the decidability and axiomatization of query finiteness in deductive databases. *JACM*, 45(4):588–633, July 1998.

[63] Michael Kifer, Raghu Ramakrishnan, and Abraham Silberschatz. An axiomatic approach to deciding query safety in deductive databases. In *PODS*, 1988.

[64] Anthony C. Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *J. ACM*, 29(3):699–717, 1982.

[65] Ravi Krishnamurthy, Raghu Ramakrishnan, and Oded Shmueli. A framework for testing safety and effective computability of extended datalog. In *SIGMOD*, 1988.

[66] Monica S. Lam, John Whaley, V. Benjamin Livshits, Michael C. Martin, Dzintars Avots, Michael Carbin, and Christopher Unkel. Context-sensitive program analysis as database queries. In *PODS*, 2005.

[67] Laurent Vieille. Recursive Axioms in Deductive Database: The Query-Subquery Approach. In *1st International Conference on Expert Database Systems*, 1986.

[68] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

[69] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, July 2006.

[70] Ondřej Lhoták. *Program Analysis using Binary Decision Diagrams*. PhD thesis, McGill University, January 2006.

[71] Senlin Liang and Michael Kifer. Deriving predicate statistics in datalog. In *PPDP*, 2010.

[72] Leonid Libkin. *Elements Of Finite Model Theory*. Springer, 2004.

[73] Changbin Liu, Lu Ren, Boon Thau Loo, Yun Mao, and Prithwish Basu. Cologne: A declarative distributed constraint optimization platform. In *VLDB*, 2012.

[74] A. Livchak. Languages for polynomial-time queries. *Computer-based modeling and optimization of heat-power and electrochemical objects*, 1992. In Russian.

[75] Boon Thau Loo, Tyson Condie, Minos Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, and Ion Stoica. Declarative Networking: Language, Execution and Optimization. In *SIGMOD*, 2006.

[76] Boon Thau Loo, Tyson Condie, Minos Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, and Ion Stoica. Declarative networking. *Commun. ACM*, 52(11):87–95, 2009.

[77] Boon Thau Loo, Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Timothy Roscoe, and Ion Stoica. Implementing Declarative Overlays. In *SOSP*, 2005.

[78] Boon Thau Loo, Harjot Gill, Changbin Liu, Yun Mao, William R. Marczak, Micah Sherr, Anduo Wang, and Wenchao Zhou. Recent advances in declarative networking. In *Fourteenth International Symposium on Practical Aspects of Declarative Languages (PADL)*, 2012.

[79] Boon Thau Loo, Joseph M. Hellerstein, Ryan Huebsch, Timo thy Roscoe, and Ion Stoica. Analyzing P2P Overlays with Recursive Queries. Technical Report UCB-CS-04-1301, UC Berkeley, 2004.

[80] Boon Thau Loo, Joseph M. Hellerstein, Ion Stoica, and Raghu Ramakrishnan. Declarative routing: extensible routing with declarative queries. In *SIGCOMM*, 2005.

[81] Boon Thau Loo, Sailesh Krishnamurthy, and Owen Cooper. Distributed Web Crawling over DHTs. Technical Report UCB-CS-04-1305, UC Berkeley, 2004.

[82] Bertram Ludäscher. *Integration of Active and Deductive Database Rules*, volume 45 of *DISDBIS*. Infix Verlag, St. Augustin, Germany, 1998. PhD thesis.

[83] Bertram Ludäscher, Ulrich Hamann, and Georg Lausen. A logical framework for active rules. In *COMAD*, 1995.

[84] Bertram Ludäscher, Wolfgang May, and Georg Lausen. Nested transactions in a logical language for active rules. In *LID*, 1996.

[85] William R. Marczak, Shan Shan Huang, Martin Bravenboer, Micah Sherr, Boon Thau Loo, and Molham Aref. Secureblox: customizable secure distributed data processing. In *SIGMOD*, 2010.

[86] Michael Meier, Michael Schmidt, and Georg Lausen. On chase termination beyond stratification. *PVLDB*, 2(1):970–981, 2009.

[87] 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.

[88] Jack Minker. Logic and databases: A 20 year retrospective. In Dino Pedreschi and Carlo Zaniolo, editors, *Logic in Databases*, volume 1154 of *Lecture Notes in Computer Science*, pages 1–57. Springer Berlin / Heidelberg, 1996. 10.1007/BFb0031734.

[89] Inderpal Singh Mumick and Hamid Pirahesh. Implementation of magic-sets in a relational database system. In *SIGMOD*, 1994.

[90] Inderpal Singh Mumick, Hamid Pirahesh, and Raghu Ramakrishnan. The magic of duplicates and aggregates. In *VLDB*, 1990.

[91] Vivek Nigam, Limin Jia, Boon Thau Loo, and Andre Scedrov. Maintaining distributed logic programs incrementally. In *13th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP)*, 2011.

[92] Orchestra Collaborative Data Sharing System. `http://code.google.com/p/penn-orchestra/`.

[93] P2: Declarative Networking System. `http://p2.cs.berkeley.edu`.

[94] Christos H. Papadimitriou. A note on the expressive power of prolog. *Bulletin of the EATCS*, 26:21–22, 1985.

[95] Lucian Popa, Yannis Velegrakis, Mauricio A. Hernández, Renée J. Miller, and Ronald Fagin. Translating web data. In *VLDB*, 2002.

[96] R. Ramakrishnan, F. Bancilhon, and A. Silberschatz. Safety of recursive horn clauses with infinite relations. In *PODS*, 1987.

[97] Raghu Ramakrishnan, Kenneth A. Ross, Divesh Srivastava, and S. Sudarshan. Efficient Incremental Evaluation of Queries with Aggregation. In *SIGMOD*, pages 204–218, 1992.

[98] Raghu Ramakrishnan, Kenneth A. Ross, Divesh Srivastava, and S. Sudarshan. Efficient incremental evaluation of queries with aggregation. In *SIGMOD*, 1994.

[99] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, and Praveen Seshadri. The CORAL deductive system. *VLDB Journal*, 3(2):161–210, 1994.

[100] Raghu Ramakrishnan and Jeffrey D. Ullman. A Survey of Research on Deductive Database Systems. *Journal of Logic Programming*, 23(2):125–149, 1993.

[101] RapidNet Declarative Networking Engine. `http://netdb.cis.upenn.edu/rapidnet/`.

[102] Thomas Reps. Demand interprocedural program analysis using logic databases. *Applications of Logic Databases*, pages 163–196, 1994.

[103] Kenneth Ross. A syntactic stratification condition using constraints. In *ILPS*, 1994.

[104] Kenneth A. Ross. Modular stratification and magic sets for datalog programs with negation. *J. ACM*, 41:1216–1266, November 1994.

[105] Kenneth A. Ross. Structural totality and constraint stratification. In *PODS*, 1995.

[106] Kenneth A. Ross and Yehoshua Sagiv. Monotonic aggregation in deductive databases. *Journal of Computer and System Sciences*, 54(1):79–97, 1997.

[107] Y. Sagiv and M. Y. Vardi. Safety of datalog queries over infinite databases. In *PODS*, 1989.

[108] Damien Sereni, Pavel Avgustinov, and Oege de Moor. Adding magic to an optimising datalog compiler. In *SIGMOD*, 2008.

[109] Praveen Seshadri, Joseph M. Hellerstein, Hamid Pirahesh, T. Y. Cliff Leung, Raghu Ramakrishnan, Divesh Srivastava, Peter J. Stuckey, and S. Sudarshan. Cost-based optimization for magic: Algebra and implementation. In *SIGMOD*, 1996.

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

[111] Divesh Srivastava and Raghu Ramakrishnan. Pushing constraint selections. In *PODS*, 1992.

[112] Leon Sterling and Ehud Shapiro. *The Art of Prolog*. The MIT Press, 2nd edition, 1994.

[113] Michael Stonebraker and Joseph M. Hellerstein, editors. *Readings in Database Systems, Third Edition*. Morgan Kaufmann, 1998.

[114] Peter J. Stuckey and S. Sudarshan. Compiling query constraints (extended abstract). In *PODS*, 1994.

[115] S. Sudarshan and Raghu Ramakrishnan. Aggregation and relevance in deductive databases. In *VLDB*, 1991.

[116] Frank Tip. A survey of program slicing techniques. *Journal of Programming Languages*, 3:121–189, 1995.

[117] Jeffrey D. Ullman. Implementation of logical query languages for databases. *ACM Trans. Database Syst.*, 10:289–321, September 1985.

[118] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems: Volume II: The New Technologies*. W. H. Freeman & Co., New York, NY, USA, 1990.

[119] M. H. Van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23:733–742, October 1976.

[120] Allen Van Gelder. The well-founded semantics of aggregation. In *PODS*, 1992.

[121] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38:619–649, July 1991.

[122] Moshe Y. Vardi. The complexity of relational query languages. In *STOC*, 1982.

[123] John Whaley and Monica S. Lam. Cloning-based context-sensitive pointer alias analysis using binary decision diagrams. In *PLDI*, 2004.

[124] Jennifer Widom and Stefano Ceri. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan-Kaufmann, 1996.

[125] Wenchao Zhou, Yun Mao, Boon Thau Loo, and Martín Abadi. Unified Declarative Platform for Secure Networked Information Systems. In *ICDE*, 2009.