Deductive Databases:

A short Overview for *The Handbook of Data Mining and Knowledge Discovery*
Oxford University Press
By: Carlo Zaniolo, CS Department, UCLA, August 1998.

Keywords: Database Languages, Rule Languages, Logic and Databases, Databases and Knowledge Bases, Expressive Power of Query Languages, Recursive Queries, Forward Chaining, Fixpoint Computation, Magic Sets, NonMonotonic Reasoning, Stratified Negation.

Introduction

Relational databases, which support simple logic-based query languages [13] were proposed in early 70s [12]. Initially considered too theoretical and not conducive to efficient implementation or commercial payoff, by mid 80s they had become a stellar success motivating research into more powerful query and data manipulation languages. A Prolog-like language, called Datalog, became the focus of this research, owing to the kinship of relational calculi and Prolog's Horn clauses [53, 60], and the surge of interest in rule-based languages for expert systems of the mid-80s. The main objectives of deductive databases are:

i) logic-based extensions of database languages to support more powerful queries and reasoning, rule-based programming, and (ultimately) achieving Turing completeness, and

ii) the efficient and scalable implementation of these declarative languages on large databases.

Language Design and Semantics

The notion of *relational completeness* is based on the fact that relational algebra expressions and safe (tuple or domain) relational calculus formulae are equivalent in terms of expressive power [13]. Thus, database query languages must be relational complete, i.e., at least as expressive as these two. Commercial languages, such as SQL, QUEL, and QBE are based on relational calculi and are relationally complete (they also support set aggregates not expressible in relational algebra [1, 60]). Now, nonrecursive safe Datalog is equivalent to relational algebra, and its expressive power can be extended by allowing *recursion* and *complex records* [53, 1, 60].

Recursion is required to express many useful queries, such as transitive closure [10]. The following Datalog rules compute the transitive closure of a directed graph stored as a binary relation $\mathtt{g(Outnode, Innode)}$:

$$\begin{aligned}\mathtt{tc(X,Z)} \leftarrow\ & \mathtt{g(X,Z).}\\ \mathtt{tc(X,Z)} \leftarrow\ & \mathtt{g(X,Y), tc(Y,Z).}\end{aligned}$$

These rules can also be viewed as defining a transformation T—in relational algebra, T is the union of the relation `g` (first rule) with the natural join of relations `tc` and `g` with the `X`-column projected out (second rule). Thus T can be first applied to the empty set, and then repeatedly on the result of the last application until no new `tc`-tuples are generated. The value of `tc` at this point is equal to the least fixpoint of T and contains the transitive closure of `g` [26, 60].

The great increase in expressive power brought by recursion is not without a price, since the fixpoint semantics and other "natural" semantics no longer hold when negation or other nonmonotonic constructs are used in the program [26, 27]. Negation in databases and in knowledge bases is nonmonotonic because negative information is not stored explicitly, rather, it is derived by default [27] under a closed-world assumption [39, 28]. Moreover, the notion of stable models [14] that provides a sound declarative semantics for programs with negated goals cannot be used directly as operational semantics because of its exponential complexity [47, 27].

Therefore, the identification of classes of Datalog programs whose stable models can be computed in polynomial time has been the focus of much research in deductive databases [60]. The notion of stratified programs (where negation can be used on predicates defined in lower strata, but not on the predicate defined by the same rule) represents a milestone in this effort [35]. In fact, programs with stratified negation can express all queries that can be computed in polynomial time in the size of the database [1]. This conclusion, however, only holds under the assumption that the underlying universe is totally ordered—as needed, e.g., to compute the count aggregate by enumerating the elements of a set one by one. A better solution is the nondeterministic construct *choice* that enforces functional dependencies in derived relations[46, 15]. While choice can be defined using negation under stable model semantics, it generates monotonic transformations; thus, it can be freely used in recursive rules [61], along with online aggregates [4], since these can be expressed using choice [61]. In general, aggregates are nonmonotonic and subject to the same restrictions as negation (modulo exceptions [57, 41]).

Important classes of programs with nonstratified negation where stable models can be computed in polynomial time include locally stratified programs [35], whose well-founded models [56, 40] can be derived by an alternating fixpoint procedure [55], and XY-stratified programs [59, 22].

Most Datalog languages also support functors, which are used to (i) store complex objects in the database, and (ii) in recursive rules to process structures of unbound length, as to achieve Turing completeness in the language. Extensions of complex objects are supported in deductive object-oriented databases [24].

Compilation and Optimization

Unlike Prolog, deductive databases draw a sharp distinction between facts (extensional information) and rules (intensional information) [28]. The rules are treated as program, and thus compiled and executed against the database which contains a time-varying set of facts described by a time-invariant schema. The target for this compilation is code based on relational algebra operators or "get next tuple" kind of operators [11, 33, 60].

Sophisticated compilation techniques are used for recursion. The naive fixpoint computation consists of repeated firing of the rules till saturation, using a table to memorize new results without duplicates. This forward-chaining, bottom-up computation is then optimized by minimizing repetitions between steps. For example, in the transitive closure computation of `tc`, only the Δ`tc` produced in the last step need be considered in the following step. Symbolic differentiation techniques are used by the compiler to realize this improvement, yielding the differential fixpoint (a.k.a. seminaive fixpoint) computation [8, 6, 45].

The differential fixpoint computes efficiently all pairs of nodes in the closure. But when only nodes connected to a given node `a` are of interest (i.e., the query goal is `tc(a,W)`), then a top-down search, such as Prolog's backward chaining, can be much more efficient [26]. Deductive databases address this problem by rule-rewriting methods that transform the original program into one that is specialized for this specific query. For the query goal `tc(a,W)`, our transitive-closure program is specialized by letting `a` become the first argument in every occurrence of `tc` (i.e., each `X` is replaced by `a`). However, if the query goal is `g(W,a)`, then this naive specialization approach does work and a left-to-right linear rule transformation method is used instead [31], producing the following program:

$$\begin{array}{ll} \mathtt{tc(X,a)} \leftarrow & \mathtt{g(X,a).} \\ \mathtt{tc(X,a)} \leftarrow & \mathtt{g(X,Y),tc(Y,a).} \end{array}$$

More sophisticated methods are needed for more general recursive rules, and many methods have been proposed for different situations [34, 19, 43, 58, 44, 20, 21, 5, 18]; the most robust and widely used method is *magic sets* [7, 9, 42]. As in the previous examples, a different execution plan is generated for each query—actually, for each template $\mathtt{tc}^{\mathtt{bf}}$ describing the goal `tc(a,X)` and all goals where the first argument is bound and the second is free. Compilers use these bound/free adornments to select recursive methods and execution orders for rule goals as to ensure safe and optimized executions [33, 53, 60].

Systems and Applications

Many interesting applications of deductive databases have been reported, ranging from the rapid prototyping of information systems and scientific databases, to data cleaning and stock market analysis [30, 51, 37]. More recent applications include websearching [25], integration and mediation of heterogeneous information systems [32, 2], and GUI generation [17].

Deductive database languages support rapid prototyping via rules that express both domain knowledge (either user-encoded or discovered) and database searches; these traits make them well-suited for data mining and knowledge discovery applications [50, 23, 3, 48, 49, 52, 16]. The main obstacle remains the lack of commercial systems. Indeed, while powerful prototypes are available [11, 36, 54], they have not yet matured into commercial systems. These and other research prototypes are surveyed in [38].

References

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

[2] R. Jr Bayardo, W. Bohrer, R. Brice et al. InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environment. In *Proceedings of the International ACM SIGMOD Conference on Management of Data*, pp. 195-206, 1997.

[3] J. Han, Y. Huang, N. Cercone, and Y. Fu. Intelligent Query Answering by Knowledge Discovery Techniques, *IEEE Transactions on Knowledge and Data Engineering*, 8(3):373-390, 1996.

[4] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online Aggregation." In *Proceedings of ACM-SIGMOD Conference on Management of Data*, May 1997.

[5] H. Aly and Z. M. Özsoyoğlu. Synchronized Counting Method. In *Proceedings IEEE International Conference on Data Engineering*, pp. 366–373, 1989.

[6] I. Balbin and K. Ramamohanarao. A Generalization of the Differential Approach to Recursive Query Evaluation. *Journal of Logic Programming* 4(3):259–262, 1987.

[7] F. Bancilhon, D. Maier, Y. Sagiv, and J. D. Ullman. Magic Sets and Other Strange Ways to Implement Logic Programs. In *Proceedings ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 1–15, 1986.

[8] R. Bayer. Query Evaluation and Recursion in Deductive Database Systems. Technical report, Technical University of Munich, Germany, 1985.

[9] C. Beeri and R. Ramakrishnan. On the Power of Magic. In *Proceedings ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 269–283, 1987.

[10] A. Chandra and D. Harel. Computable Queries for Relational Data Bases. *Journal of Computer and System Sciences*, 21(2), 1980.

[11] D. Chimenti, R. Gamboa, R. Krishnamurthy, S. Naqvi, S.Tsur and C. Zaniolo. The LDL System Prototype. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), 76-90 (1990). *A web demo of* $\mathcal{LDL}$*++ is available at* `http://www.cs.ucla.edu/ldl`.

[12] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the Association of Computing Machinery* 13(6):377–387, 1970.

[13] E. F. Codd. Relational Completeness of Database Sublanguages. In R. Rustin (ed.), *Data Base Systems.* Prentice Hall, Englewood Cliffs, NJ, 1972.

[14] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. *Proceedigns of Joint International Conference and Symposium on Logic Programming*, pp. 1070–1080, Seattle, WA, 1988.

[15] F. Giannotti, D. Pedreschi, D. Saccà, and C. Zaniolo. Non-Determinism in Deductive Databases. In C. Delobel, M. Kifer, and Y. Masunaga (eds.), *Proceedings Deductive and Object-Oriented Databases, Second International Conference, DOOD'91*, pp. 129–146, Springer-Verlag, New York, 1991.

[16] F. Giannotti, G. Manco, M. Nanni, D. Pedreschi, Using Deduction for Intelligent Data Analysis, CNUCE-CNR technical report, Pisa, Italy, 1998.

[17] N. Goyal, C. Hoch, R. Krishnamurthy, B. Meckler and M. Suckow. Is GUI programming a database research problem? *Proc. Int. ACM SIGMOD Conference on Management of Data*, pp. 517-528, 1996.

[18] S. Greco and C. Zaniolo. Optimization of Linear Logic Programs Using Counting Methods. In *Proceedings International Conference on Extending Database Technology*, 1992.

[19] L. Henschen and S. Naqvi. On Compiling Queries in Recursive First-Order Databases. *Journal of the ACM*, 31(1):47–85, 1984.

[20] R. Haddad and J. F. Naughton. Counting Methods for Cyclic Relations. In *Proceedings ACM SIGACT-SIGMOD-SIGART Principles of Database Systems*, pp. 333–340, 1988.

[21] J. Han. Selection of Processing Strategies for Different Recursive Queries. In *Proceedings International Conference on Data and Knowledge Bases*, Jerusalem, Israel, 1988.

[22] D. Kemp, K. Ramamohanarao, and P. Stuckey. ELS Programs and the Efficient Evaluation of Non-Stratified Programs by Transformation to ELS. In *Proceedings Int. Conf. on Deductive and Object-Oriented Databases: DOOD95*, pp. 91–108, Singapore, December 1995.

[23] B. Kero, L. Russell, S. Tsur and W.M. Shen, An overview of Data Mining Technologies, DOOD95 Workshop on the Integration of Knowlege Discovery Deductive and Object Oriented Databases, Singapore, 1995.

[24] M. Kifer. Deductive and Object-Oriented Data Languages—A Quest for Integration. *Proceedings Int. Conf. on Deductive and Object-Oriented Databases, DOOD'95*, pp. 187-212, 1995.

[25] L.V.S. Lakshmanan, F. Sadri, and I. Subramanian. A Declarative Approach to Querying and Retsructuring the World-Wide-Web. Post-ICDE Workshop on Research Issues in Data Engineering (RIDE'96). New Orleans, February 1996.

[26] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, New York, 1987.

[27] V. W. Marek and M. Truszczynski. *Nonmonotonic Logic.* Springer-Verlag, New York, 1995.

[28] J. Minker. Logic and Databases: A 20 Year Retrospective. In D. Pedreschi and C. Zaniolo (eds.), *Proceedings International Workshop on Logic in Databases (LID'96)*, Springer-Verlag, New York, pp. 5–52, 1996.

[29] K. Morris, J. D. Ullman, and A. Van Gelder. Design Overview of the NAIL! System. In *Proceedings International Conference on Logic Programming*, pp. 554–568, Springer-Verlag, LNCS 225, New York, 1986.

[30] S. Naqvi and S. Tsur. *A Logic Language for Data and Knowledge Bases.* Computer Science Press, New York, 1989.

[31] J. F. Naughton. Compiling Separable Recursions. In *Proceedings ACM International Conference on Management of Data*, pp. 312–319, 1988.

[32] Y. Papakonstantinou, H. Garcia-Molina, J.D. Ullman. MedMaker: A Mediation System Based on Declarative Specifications. *Proceedings IEEE Int. Conference on Data Engineering*, pp. 132-141, 1996.

[33] G. Phipps, M. A. Derr, and K. A. Ross. Glue-Nail!: A Deductive Database System. In *Proceedings International ACM SIGMOD Conference on Management of Data*, pp. 308–317, 1991.

[34] F. C. N. Pereira and D. H. D. Warren. Parsing as Deduction. In *Proceedings Annual Meeting Association for Computational Linguistics*, pp. 137–144, 1983.

[35] T. C. Przymusinski. On the Declarative and Procedural Semantics of Stratified Deductive Databases. In J. W. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 193–216, Morgan Kaufman, San Francisco, CA, 1988.

[36] R. Ramakrishnan, D. Srivastava, S. Sudanshan, and P. Seshadri. Implementation of the CORAL Deductive Database System. In *Proceedings of the International ACM SIGMOD Conference on Management of Data*, pp. 167–176, 1993.

[37] R. Ramakrishnan. *Applications of Logic Databases.* Kluwer Academic, Norwell, MA, 1995.

[38] R. Ramakrishnan and J. D. Ullman. A Survey of Research in Deductive Database Systems. *Journal of Logic Programming* 23(2):125–149, 1995.

[39] R. Reiter. On Closed World Data Bases. In J. W. Minker and H. Gallaire (eds.), *Logic and Databases*, pp. 55–76, Plenum Press, New York, 1978.

[40] K. A. Ross. Modular Stratification and Magic Sets for Datalog Programs with Negation. *Journal of ACM* 41(6):1216–1266, 1994.

[41] K. A. Ross and Yehoshua Sagiv, "Monotonic Aggregation in Deductive Database", *JCSS*, 54(1), 79-97 (1997)

[42] D. Saccà and C. Zaniolo. Implementation of Recursive Queries for a Data Language based on Pure Horn Logic. *Proceedings Fourth Int. Conference on Logic Programming*, pp. 104–135, MIT Press, Cambridge.

[43] D. Saccà and C. Zaniolo. Magic Counting Methods. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pp. 49–59, 1987.

[44] D. Saccà and C. Zaniolo. The Generalized Counting Method of Recursive Logic Queries for Databases. *Theoretical Computer Science*, pp. 187–220, 1988.

[45] D. Saccà and C. Zaniolo. Differential Fixpoint Methods and Stratification of Logic Programs. In *Proceedings International Conference on Data and Knowledge Bases*, 1988.

[46] D. Saccà and C. Zaniolo. Stable Models and Non-Determinism in Logic Programs with Negation. In *Proceedings ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1990.

[47] J. S. Schlipf, A Survey of Complexity and Undecidability Results in Logic Programming, *Proc. Workshop on Structural Complexity and Recursion-Theoretic Methods in Logic Programming*, 1993, pp.143–164.

[48] W. Shen, K.Ong, B. Mitbander and C. Zaniolo, Metaqueries for Data Mining, Chapter 15 of *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad et al (eds.), MIT Press, 1996.

[49] W. M. Shen and Bing Leng. A Metapattern-Based Automated Discovery Loop for Integrated Data Mining. *IEEE Transactions on Data and Knowledge Engineering*, vol. 8, no. 6, pp. 898–910, 1996.

[50] S. Tsur. Data Dredging. *Data Engineering Bulletin*, 13(4): 58-63 (1990)

[51] S. Tsur. Deductive Databases in Action. In *Proceedings ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Programming Languages*, pp. 142–154, 1991.

[52] D. Tsur, J. D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, and S. Nestorov. Query Flocks: A Generalization of Association-Rule Mining. *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data*, pp. 1-12, June 1998.

[53] J. D. Ullman. *Principles of Data and Knowledge-Base Systems*, volume 3. Computer Science Press, New York, 1988.

[54] J. Vaghani, K. Ramamohanarao, D. B. Kemp, et. al. The Aditi Deductive Database System. *The VLDB Journal*, 3(2), pp. 245-288 (1994).

[55] A. Van Gelder. The Alternating Fixpoint of Logic Programming with Negation. In *Proceedings ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 1–10, 1989.

[56] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of ACM* 38:620–650, 1991.

[57] A. Van Gelder. Foundations of Aggregations in Deductive Databases In *Proc. of the Int. Conf. On Deductive and Object-Oriented databases*, 1993.

[58] L. Vielle. Recursive Axioms in Deductive Databases: The Query-Subquery Approach. In *Proceedings International Conference on Expert Database Systems*, pp. 253–268, 1986.

[59] C. Zaniolo, N. Arni, and K. Ong. Negation and Aggregates in Recursive Rules: the $\mathcal{LDL}$++ Approach. In *Proceedings International Conference on Deductive and Object-Oriented Databases, DOOD'93*, 1993.

[60] C. Zaniolo, S. Ceri, C. Faloutsos, R. Snodgrass, V.S. Subrahmanian and R. Zicari, *Advanced Database Systems*, Morgan Kaufmann, 1997.

[61] C. Zaniolo and H. Wang Logic-Based User-Defined Aggregates for the Next Generation of Database Systems. In *The Logic Programming Paradigm: Current Trends and Future Directions.* K.R. Apt, V. Marek, M. Truszczynski, D.S. Warren (eds.), Springer Verlag, 1998.