University of Florida
Computer and Information Sciences

An Open Architecture for Optimizing Active and Deductive Rules

Sharma Chakravarthy
Xiaohai Zhang

email: sharma@cis.ufl.edu

Tech. Report UF-CIS-TR-93-013

April 1993

(Submitted for publication)

Department of Computer and Information Sciences
Computer Science Engineering Building
University of Florida
Gainesville, Florida 32611

Contents

An Open Architecture for Optimizing Active and Deductive Rules*

Sharma Chakravarthy

Xiaohai Zhang

Database Systems Research and Development Center
Computer and Information Sciences Department
University of Florida, Gainesville, FL 32611
email: {sharma,xhai}@.cis.ufl.edu

April 5, 1993

Abstract

It is evident that non-traditional applications, such as engineering design, Air Traffic Control, situation assessment, Computer integrated manufacturing, program trading and cooperative problem solving require both deductive and active capabilities in addition to the functionality provided by a traditional DBMS. In this paper, we propose an extensible query optimizer architecture for supporting both active and deductive rules. We first discuss the similarities and differences of the optimization techniques used in deductive and active databases, analyze them and then propose an extensible optimizer to support the techniques in a uniform manner. This paper will focus primarily on the overall architecture of the optimizer, control strategies, interaction of the functional components, and their impact on the overall system.

Index Terms: Deductive databases, Active databases, Extensible architecture, Domain experts

*This work was supported, in part, by the National Science Foundation Grant IRI-9011216.

1 Introduction

Traditionally, database management systems have supported and managed large amounts of shared data (facts or extensional data). This support for the management of extensional data has resulted in systems that provide techniques for efficient retrieval of data from secondary storage. Views in relational systems have essentially been used to provide alternate users' perspectives of stored data as well as for specifying derived data.

deductive databases generalize and extend relational databases by allowing relations to be defined implicitly in terms of rules (including recursive, stratified rules etc.). They also provide a theory that supports both declarative and procedural semantics. Logic programming, on the other hand, has concentrated on exploring efficient techniques for various classes of theories and queries. Various approaches (loosely coupled,tightly coupled, and integrated) have been proposed to combine deductive rules with relational systems [dMS88, Min88, VCJ, SHP87]. Efforts such as Megalog, LDL, RDL1 and LDL++, Coral have endeavored to integrate rule processing and traditional database functionality (and object-oriented paradigm as in LDL++ and Coral++).

The need for active database functionality arise from the applications such as Air Traffic Control, situation assessment, Computer integrated manufacturing and program trading require timely response to critical situations in addition to asynchronous monitoring of situations of interest to the application without user intervention. Functionally, an active database management system monitors *conditions* triggered by *events* and if the condition evaluates to true then the *action* is executed. Events have been broadly classified [C$^+$89] into: database events (database operations, such as insert, delete), temporal events (absolute, and relative time events), external or abstract events (e.g., report arrives, start-chip-test), and complex/composite events that are constructed from primitive events (the previous 3 categories) and a set of event operators (e.g., disjunction, sequence, one-or-more). Recent work on active databases [C$^+$89, SHP88, WF90, DB90, Int90b, GJS92a, GJ91a, DPG91, MP90, SKL89, CHS93, Anw92a, CN90, Mau92, DUHK93, GrD93] and the need for supporting situation monitoring in the form of event-condition-action rules has prompted a fresh look at the architecture of database systems [DB90, CN90, Cha89, CNG$^+$90, KDM88]. The need for integrating active functionality with traditional databases has resulted in the identification of new components (e.g., situation monitor), extensions to existing components (e.g., transaction manager) as well as new results on several issues.

It is evident that non-traditional applications, such as concurrent engineering, process control, situation assessment network management, and cooperative problem solving require both deductive and active capabilities in addition to the functionality provided by a traditional DBMS. Figure 1 summarizes the current situation along the three areas of interest - traditional databases (as points along the X-axis), deductive databases (as the plane formed by the X- and Y-axis), and active databases (as the plane formed by the X- and Z-axis). In order to obtain a system that combines features from all the three areas, we need to move towards the envelope that encompasses all the three components of interest (indicated by the three arrows) by combining techniques/concepts developed individually in each of the areas. Not surprisingly, there is considerable overlap in the functionality of the systems being combined and as a result, in some cases a single module (e.g., rule processor/optimizer) in the resulting system may adequately capture the features of the individual systems. As an aside, Figure 1 also indicates the orthogonality of the model (or the paradigm, such as relational or object-oriented) with respect to active and deductive capabilities.

In this paper, we propose an extensible architecture for optimizing both deductive and active rules. We analyze the differences and commonalities of optimization mechanisms in deductive

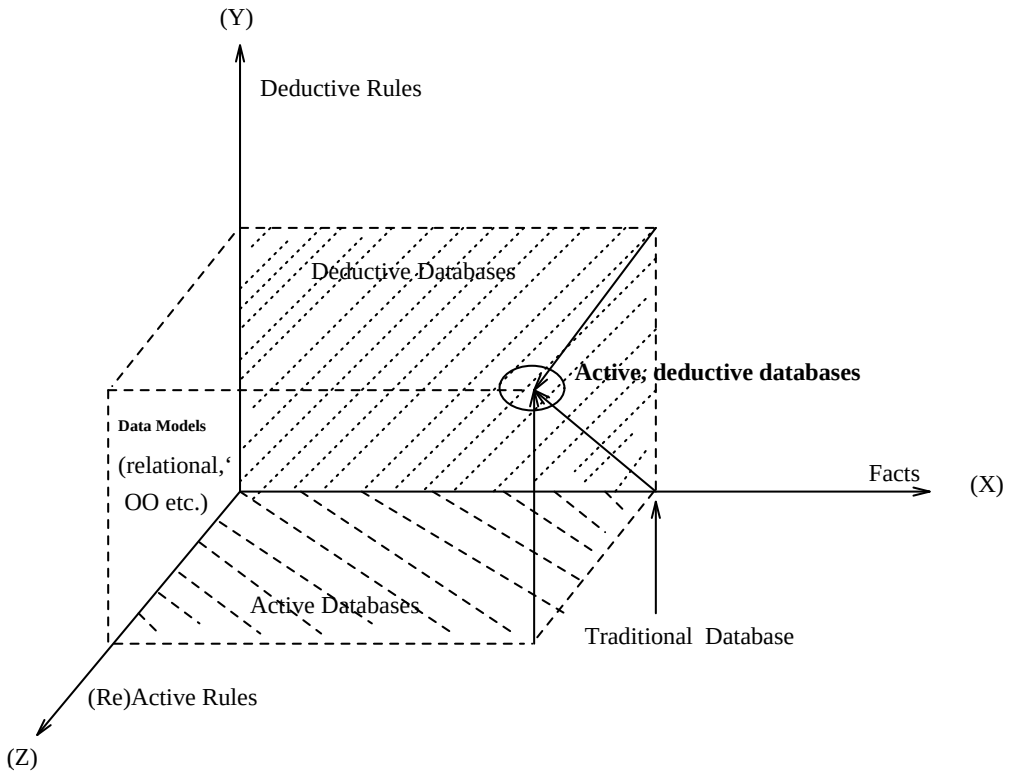

Figure 1: Space of Current Systems and Desired Goal

databases and active databases, before proposing an extensible optimizer that incorporates a set of techniques while avoiding the redundancies of commonalities. This paper will focus on the overall architecture of the optimizer, interaction of the functional components, and their impact on the overall system.

The remainder of this paper is structured as follows. Section 2 identifies the differences and commonalities among the optimization techniques used in conventional, deductive, and active databases. Section 3 proposes an open architecture based on the analysis in section 2. Section 4 describes our approach for implementing the proposed architecture. section 5 includes conclusions.

2 Analysis and Comparison of Techniques

In this section, we provide a brief overview of optimization techniques used in deductive and active databases. The discussion encompasses nonrecursive query processing and recursive query processing in deductive databases and situation evaluation techniques in active databases. Finally, we point out the differences and commonalities of these techniques which has a bearing on the design of an optimizer for the deductive and active rules.

For processing queries in nonrecursive deductive databases, conventional query optimization need only be augmented with a compilation phase to transform intensional relations in terms of extensional relations [CGM88, CGM90]. The input query is first modified using the compiled (intensional) database, and then optimized using conventional query optimization techniques. As

for recursive query processing in deductive database, numerous strategies have been proposed [BR86, LV89, Sto90, WF90]. In general,a recursive query is evaluated in the following way [BR86]:

1. Generate the intermediate result set,
2. Expand the set by evaluating the sentences within it,
3. Check the termination Condition at each step.

According to the way in which the language is generated,the strategies can be categorized into: bottom-up and top-down. The bottom-up strategies generate the answer to query starting from the base relations, while the top-down strategies produce the answer starting from the query itself. The optimization is performed to make the rules to be evaluated more efficiently. Rules or queries are transformed into equivalent rules or queries by rewriting rules or introducing new rules. For example, The Aho-Ullman algorithm rewrites recursive queries by commuting selections with the least fixpoint operator [AU79]; the Magic Sets algorithm reduces the number of potentially relevant facts by the introduction of new rules [B$^+$86].

In active databases, efficient management of event-condition-action (ECA) rules is the main concern. This translates into efficient detection of events as well as evaluation of the query (condition) and the action (transaction) associated with that event. As the situation (event+condition) optimization is typically done by an optimizer, we will restrict our discussion to situation evaluation.

Most of the work on active database systems [C$^+$89, DBAB$^+$88, RCBB89, SR86, SHP87, DB90, KDM88, WCB91, WF90, GJ91b, GJS92b, Han89, Int90a, Anw92b, CM91, CHS93, GrD93, DPG91] is aimed at supporting some form of rule processing capability (e.g., alerters, triggers, situation-action rules) and techniques for their management and optimization (e.g., lazy, eager, overlapped execution). Active capability is viewed as a unifying mechanism for supporting a number of DBMS functionality, such as integrity/security enforcement, maintenance of materialized (e.g., view) data, constraint management, and rule-based inferencing.

In contrast to deductive rules and traditional queries, in active databases, the set of all *event-condition-action rules* is likely to form a potentially large set of predefined queries that need to be efficiently managed and evaluated when specified events occur. Rule evaluation imposes an overhead on (possibly) every event. Also, when the number of rules is large, grouping related rules (rules that have common subexpressions in events and conditions, for example) to reduce computation is beneficial.

In addition to the development of new techniques, extensions to existing techniques have been proposed for optimizing ECA rules. First, ECA rules are temporally persistent. That is, they have a longer life-span and as a result are likely to be evaluated many times. This suggests that several rules can be optimized simultaneously in a group, possibly using some of the techniques developed for multiple query optimization [Fin82, CM86, RC88, Sel86, Cha91]. The effect of multiple query optimization can be further enhanced by materializing intermediate results (e.g., common subexpressions) judiciously. Second, rules used for some applications are likely to have priorities or timing requirements associated with their execution. Optimization of such rules requires different techniques, such as exhaustive optimization, novel buffering strategies, use of main memory, and appropriate processing techniques (e.g., use of parallelism).

In HiPAC [RCBB89, C$^+$89], incremental operators, a chain rule (which was generalized in [CG91]), and a signal graph was proposed for optimizing situations. The chain rule and incremental forms are similar to the transformations performed by a conventional query optimizer (e.g.,

commuting the selection with join) but on new operators. This can be identified as one of the commonalities among the techniques used in deductive and active databases. In addition, conventional optimization is required for both cases. On the other hand, the compilation phase of nonrecursive query processing in deductive databases is different from its counterpart in the situation evaluation of active databases. In addition, the transformation strategies used in recursive query processing also differ from those used for situation evaluation. Finally, another way to contrast active and deductive databases is that active databases support event driven forward chaining of rules where as deductive databases typically use the backward chaining of rules to compute answers.

3 The Cooperating Experts Architecture

Current research on query processing in databases is addressing extensibility – at the architecture, at the data model, and at the functionality level. This is in contrast to the earlier concentration on optimizing a small set of operators (e.g., relational) and incorporating them into a monolithic optimizer.

Extensible query optimizers try to alleviate the limitations of traditional approach by decoupling the optimizing engine from the knowledge that is used to optimize a query. Ideally, an extensible optimizer should be domain independent consisting of an optimizing engine that uses a repertoire of knowledge required for the purpose of optimization. The optimizing engine should preferably use an efficient and expressive representation (e.g., AND/OR graphs) and a search strategy that can be readily combined with the generation of the strategy space (e.g., heuristic state space search) which can be fine-tuned [RC88]. It should also be possible to specify the optimization parameters in a general fashion (such as CPU cost, I/O cost, the number of processors that can be used for processing a query, special purpose processors available etc.).

Three basic approaches have been used for developing extensible query optimizers: the generator approach [GD87, Mck92], the building block approach [Bat88], and the modular approach [Loh88, M+89, SR86]. Some approaches use rules for expressing transformations and user-definable cost estimates for the purpose of pruning the search space. Although conceptually it is useful to think of strategy space generation and search as separate problems, efficiency aspects of optimization dictate that they be considered together. This is evident in most of the work on extensible query optimization.

It is our belief that the separation of optimizing engine from the knowledge used to optimize a query is an important first step towards extensibility in the most general sense. When there is a need for combining functionality (as the one we are discussion now), in addition to this two-level separation, it is critical that the domain knowledge itself is partitioned (or grouped) into domain experts responsible for a specific class of optimizations. The conceptual architecture of an optimizer that supports the above is shown in figure 2. The architecture presented in this paper is similar to the philosophy of open systems (e.g., Open OODB Toolkit of Texas Instruments, Dallas [WBT92]).

Our approach to extensible query processor consisting of: an expert/intelligent coordinator, a set of domain experts each responsible for transforming a query using their domain specific expertise, and a global optimizer/assemble that integrates individual optimizations performed on the query or fine tunes the resulting query produced by the optimizer. The domain experts, in a sense, cooperate in performing the optimization of a query with the help of the intelligent coordinator. In this approach it is possible to reveal partial optimizations to other experts, if that is the control strategy used by the coordinator. This can be done by posting partial optimizations

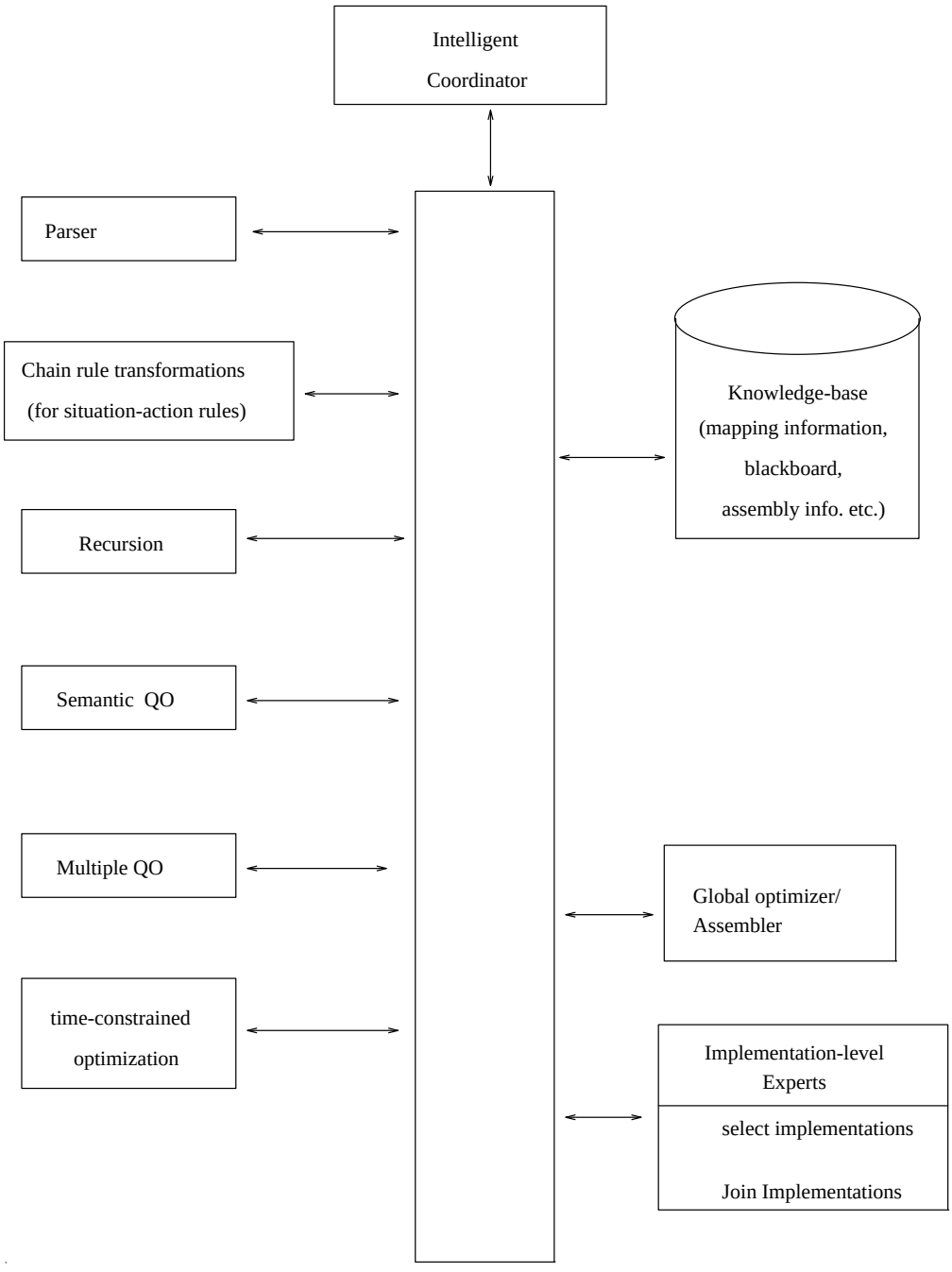

Figure 2: Conceptual Architecture of the proposed optimizer

on to a knowledge base (which acts as a persistent blackboard) to make it available to others. Also, it is less critical as to whether the individual experts are rule-based or hard-coded. A uniform representation, however, is extremely useful to eliminate translation from one representation to another before a domain expert can apply a set of transformations.

The approach presented in Figure 2 combines and further generalizes the three approaches found in the literature. Here the optimizing engine is decoupled from the knowledge used for optimization at least at two levels: at the individual domain experts level and at the overall query processor level. Further, the optimization units are modular and are based on the semantics of the activity they perform. Finally, plug compatibility is provided either using a common representation or adding a translator that is responsible for translating one representation (used by the new expert) into the uniform representation. The local experts using the same representation can share a translator, otherwise, each of them need to have a different translator. The generalization is in terms of the choice of control strategies at the individual (local expert) level as well as the global level; the generator as well as traditional approach can be used to generate individual experts or the intelligent coordinator.

The approach presented above is intended to accommodate:

1. New data types and their operations (spatial data, time): by adding a new local expert or enhancing an existing local implementation expert (and adding cost models to the knowledge base),
2. Special optimization (e.g., recursion): by again enhancing, adding a new local expert,
3. new environments (e.g., multiprocessors, specialized components): by changing the control strategy of the coordinator as well as architectural knowledge required for query optimization, and
4. New optimization approaches (e.g., semantic optimization, chain-rule transformations): by adding new local experts.

Finally, the approach presented here can also be used to optimize queries in several different ways and compare them. For example, query-subquery approach for recursive processing can be compared with magic sets by allowing the same query to be optimized by two local experts. The common transformations can still be used once without having to replicate them. This capacity can be important in cases where the choice of optimization strategy is dependent on the query class which is linked to application semantics.

3.1 Intelligent Coordinator

The intelligent coordinator plays a critical role in this architecture. The coordinator has the knowledge of the availability of local experts and coordinates their usage to perform optimization. Some of the activities (some of these can be relegated to a local expert) of the coordinator include parsing, rewriting, decomposing, distributing and collecting optimized sub-queries and globally optimizing the query to produce the final result.

The **Knowledge Base** contains information for parsing, rewriting, decomposing and distributing a query. The knowledge base also provides statistical information used by the implementation

experts. The knowledge base includes information about the local experts in the optimizer, global optimization strategies, etc. They are kept current as the environment changes.

Indeed, the control strategy used by the coordinator can simulate different sequencing of local experts and even accommodate different architectures on which this approach to optimization is being implemented. We illustrate three possible control strategies:

Centralized: Under this control strategy, the intelligent coordinator is responsible for sequencing the operations among local experts. In other words, the coordinator decides the sequence in which the input query is optimized by local experts. The coordinator may also perform some global optimization before invoking a local expert or even invoke a local expert more than once.

decentralized/Parallel: Under this scheme, the coordinator decomposes the input query and send it to appropriate local experts. The optimization on the decomposed query is performed in a totally decentralized manner. The results of each expert is collected and assembled into the final version either by the coordinator or an assembler using global knowledge. It is also possible for the input query to be distributed to local experts without decomposition, and the local experts will extract the subquery on which they can perform optimization. However, this strategy will introduce additional difficulty when partial optimizations are assembled to produce the final version.

Cooperating experts: This strategy combines the previous two in that local experts perform partial optimization and can share partial results through the shared knowledge base. Under this scheme each alternative query produced by a local expert can be made available to others. Pruning of alternatives as well as assembly of individual plans to obtain a final plan need to be performed. This strategy requires a high degree of mediation from the coordinator for it to be successful.

When designing the architecture of optimizer,there needs a balance between the centralized approach and decentralized approach. Both of the strategies have advantages and disadvantages: centralized approach is much easier to implement and make extensions,but it is not feasible for a heterogeneous system and the coordinator may take too muck work; decentralized strategy fits a larger range of systems and avoids the overhead of coordinator,but it is more difficult to implement. For a particular system,it is reasonable to take parts from the above strategies and merge them into an efficient optimizer.

3.2 Issues

There are a number of issues that need to be addressed in order to realize the architecture presented above. Some of them are discussed below:

Interface: As discussed in [RC88], an expressive canonical intermediate representation is critical for the success of this approach. In the absence of it, transformations have to be performed from one representation to the other incurring substantial overhead and difficulty in sharing partial results. Although modules (or local experts) for transformation can be easily included in the framework, the system becomes extremely complicated as the complexity grows quadratically with the number of local experts. It is conceivable that groups of local experts will have a common representation and some transformation may have to be performed.

We propose an AND/OR graph as the canonical representation as it generalizes the operator trees (that is widely used) and allows for the representation of multiple strategies in the representation.

Search: The search problem is to find the cheapest query processing strategy from among the alternatives that are available. Although it makes sense to conceptually separate strategy space generation and search, practically they are fairly intertwined to reduce the complexity of search. This is a serious problem in the current approach as the strategy space is produced for small portions of a query and not enough global information is available.

Several approaches are possible. Local pruning can be performed by individual experts hoping for the best. Or instead of choosing a strategy, a small number of strategies can be obtained by each local expert which are further evaluated by the global optimizer/assembler. It may also be possible to reveal some cost information to each local experts (based on the control strategy used by the coordinator) which can be used to make local decisions. These alternatives need further evaluation.

Coordinator: The coordinator is a critical component of this approach. It is the glue that encodes the control strategy as well as the information that is passed to local experts (e.g., required for search as explained above). It is also responsible for updating the knowledge base that stores partial results, global heuristics, and statistics.

4 Implementation

Currently, at the University of Florida, we are developing an open query processor as part of the Sentinel project. We are using the Open OODB Toolkit (Alpha release from texas Instruments, Dallas [WBT92]) as the underlying platform for this effort. We intend to concentrate on the open architecture and the issues mentioned above and plan on using available tools where possible.

Our short immediate term goal is to use the Volcano [Mck92, BMG92] optimizer generator (by modifying it where necessary) to move towards the open architecture. Initially, we will try to group rules into modules that roughly correspond to local experts. This requires modifications to relevant parts of Volcano to match our requirements.

As an intermediate term goal, we plan to generate a few local experts using the modified Volcano optimizer generator and build the coordinator with a pre-defined control strategy. Currently, Volcano uses a fixed control strategy for the optimizer it generates.

Our long term goal is to build an optimizer generator that generates local experts as well as the coordinator for a given choice of coordinator control strategy. We plan on conducting experiments using the first immediate and intermediate systems prior to determining the implementation approach for the long term goal.

Finally, we would like to point out that the proposed architecture supports a migration path between pre-existing systems and the new one. Parts of current optimizers can be easily converted into local experts by adding an interface that will translate the current format into the uniform representation without changing the function itself. New local experts can be developed on a demand basis to conform to the new representation used. Of course, it is difficult to map an existing search strategy (such as the one in System R, for example) on to this architecture without some difficulty.

5 Conclusions

In this paper, we propose the open architecture for optimizing queries in deductive-active database systems. In fact, the architecture is a general purpose one and can be used in several environments where there is a need for combining the functionality of individual systems. The open architecture allows one to match the requirements of the resulting system by adding relevant components (i.e., local experts) and removing unwanted components from the optimizer with little effort. In our view, such a query optimizer is highly desirable for rapidly emerging new systems.

We also envision the use of such optimizers in heterogeneous environments where a simpler form of the intelligent coordinator is required. In contrast to the approaches proposed, in this case, global optimization (including the way in which the results are assembled) is performed prior to decomposition using some knowledge of the systems involved.

We are currently investigating the various issues presented in this paper. In the short term, we propose to generate a few local experts using the Volcano optimizer generator and build the coordinator. Our long term plans include a generator approach for obtaining the coordinator.

References

[Anw92a] E. Anwar. Supporting complex events and rules in an oodbms: A seamless approach. Master's thesis, Database Systems R&D Center, CIS Department, University of Florida, E470-CSE, Gainesville, FL 32611, November 1992.

[Anw92b] E. Anwar. Supporting complex events and rules in an oodbms: A seamless approach. Master's thesis, Database Systems R&D Center, CIS Department, University of Florida, E470-CSE, Gainesville, FL 32611, November 1992.

[AU79] A. Aho and J. Ullman. Universality of data retrieval languages. In *Proc. of the 6th ACM Symposium on Principles oof Programming Languages*, 1979.

[B+86] F. Bancilhon et al. Magic sets: Algorithms and examples. *Unpublished manuscript*, 1986.

[Bat88] D. Batory. GENESIS: An Extensible Database Management System. *IEEE Transactions on Software Eng*, Nov. 1988.

[BMG92] Jose A. Blakeley, William J. Mckenna, and Gectz Graefe. Experiences building the open oodb query optimizer. Technical report, Texas Instruments INC., 1992.

[BR86] F. Bancilhon and R. Ramakrishnan. An Amateur's Introduction to Recursive Query Processing Strategies. In *Proceedings ACM SIGMOD Conference on Management of Data*, pages 16–52, 1986.

[C+89] S. Chakravarthy et al. HiPAC: A Research Project in Active, Time-Constrained Database Management, Final Report. Technical Report XAIT-89-02, Xerox Advanced Information Technology, Cambridge, MA, Aug. 1989.

[CG91] S. Chakravarthy and S. Garg. Extended relational algebra (era): for optimizing situations in active databases. Technical Report UF-CIS TR-91-24, Database Systems R&D

Center, CIS Department, University of Florida, E470-CSE, Gainesville, FL 32611, Nov. 1991.

[CGM88] S. Chakravarthy, J. Grant, and J. Minker. Foundations of semantic query optimization for deductive databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 243–273. Morgan Kaufmann Publications, 1988.

[CGM90] S. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, 1990.

[Cha89] S. Chakravarthy. Hipac: A research project in active,time-constrained database management. Technical report, Xerox Advanced Information Technology, 1989.

[Cha91] S. Chakravarthy. Divide and Conquer: A Basis for Augmenting a Conventional Query Optimizer with Multiple Query Processing Capabilities. In *Proc. of the 7th Int'l Conf. on Data Engineering, Kobe, Japan*, pages 482–490, Apr. 1991.

[CHS93] S. Chakravarthy, E. Hanson, and S.Y.W. Su. Active Database Research at the University of Florida. *To appear in IEEE Quarterly Bulletin on Data Engineering*, January 1993.

[CM86] U. S. Chakravarthy and J. Minker. Multiple Query Processing in Deductive Databases Using Query Graphs. In *Proceedings of International Conference of Very Large Data Bases*, pages 384–391, 1986.

[CM91] S. Chakravathy and D. Mishra. An event specification language (snoop) for active databases and its detection. Technical Report UF-CIS TR-91-23, Database Systems R&D Center, CIS Department, University of Florida, E470-CSE, Gainesville, FL 32611, Sep. 1991.

[CN90] U. S. Chakravarthy and S. Nesson. Making an Object-Oriented DBMS Active: Design, Implementation and Evaluation of a Prototype. In *Proc. of Int'l Conf. on Extended Database Technology (EDBT), Kobe, Japan*, pages 482–490, Apr. 1990.

[CNG+90] S. Chakravarthy, S. B. Navathe, S. Garg, D. Mishra, and A. Sharma. An evaluation of active dbms developments. Technical Report UF-CIS TR-90-23, Database Systems R&D Center, CIS Department, University of Florida, E470-CSE, Gainesville, FL 32611, Sep. 1990.

[DB90] M. Darnovsky and J. Bowman. *TRANSACT-SQL USER'S GUIDE, Release 4.2.* Document 3231-2.1, Sybase Inc., May 1990.

[DBAB+88] U. Dayal, B. Blaustein, S. Chakravarthy A. Buchmann, et al. The HiPAC project: Combining active databases and timing constraints. *Special Issue of Real Time Data Base Systems, SIGMOD Record*, 17(1):51–70, Mar. 1988.

[dMS88] C. de Mairdreville and E. Simon. Modelling queries and updates in deductive databases. In *proc. of the 14th Int'l Conf. on VLDB,*, Los Angles, August 1988.

[DPG91] O. Diaz, N. Paton, and P. Gray. Rule Management in Object-Oriented Databases: A Unified Approach. In *Proceedings 17th International Conference on Very Large Data Bases*, Barcelona (Catalonia, Spain), Sept. 1991.

[DUHK93] S. W. Dietrich, S. D. Urban, J. V. Harrison, and A. P. Karadimce. A DOOD RANCH at ASU: Integrating Active, Deductive and Object-Oriented Databases. *To appear in IEEE Quarterly Bulletin on Data Engineering*, January 1993.

[Fin82] S. Finkelstein. Common Expression Analysis in Database applications. In *Proc. of ACM-SIGMOD, Orlando*, Jun. 1982.

[GD87] G. Graefe and D. DeWitt. The Exodus Optimizer Generator. In *Proc. of ACM-SIGMOD*, pages 160–172, 1987.

[GJ91a] N. H. Gehani and H. V. Jagadish. Ode as an Active Database: Constraints and Triggers. In *Proceedings 17th International Conference on Very Large Data Bases*, pages 327–336, Barcelona (Catalonia, Spain), Sep. 1991.

[GJ91b] N. H. Gehani and H. V. Jagadish. Ode as an Active Database: Constraints and Triggers. In *Proceedings 17th International Conference on Very Large Data Bases*, pages 327–336, Barcelona (Catalonia, Spain), Sep. 1991.

[GJS92a] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Event Specification in an Object-Oriented Database. In *Proceedings International Conference on Management of Data*, pages 81–90, San Diego, CA, June 1992.

[GJS92b] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Event Specification in an Object-Oriented Database. In *Proceedings International Conference on Management of Data*, pages 81–90, San Diego, CA, June 1992.

[GrD93] S. Gatziu and K. r. Dittrich. SAMOS: an Active, Object-Oriented Database System. *To appear in IEEE Quarterly Bulletin on Data Engineering*, January 1993.

[Han89] Eric N. Hanson. An Initial Report on the Design of Ariel: a DBMS with an integrated production rule system. *ACM SIGMOD RECORD*, 18(3):12–19, Sep. 1989.

[Int90a] Interbase Software Corporation, 209 Burlington Road, Bedford, MA 01730. *DDL Reference*, February 1990.

[Int90b] InterBase Software Corporation, Bedford, MA. *InterBase DDL Reference Manual, InterBase Version 3.0*, 1990.

[KDM88] A. M. Kotz, K. R. Dittrich, and J. A. Mulle. Supporting Semantic Rules by a Generalized Event/Trigger Mechanism. In *Proceedings International Conference on Extended Data Base Technology*, Venice, Mar. 1988.

[Loh88] G. M. Lohman. Grammer-like functional rules for representing query optimizationalternatives. In *Proc. of ACM-SIGMOD*, pages 18–27, 1988.

[LV89] A Lefebvre L Vieille. Deductive database systems and the dedgin query evaluator. In *proc. of the 7th British National Conference on Database*, Heriot-Watt University, July 1989.

[M+89] L. M.Haas et al. Extensible query processing in starburst. *ACM-SIGMOD*, pages 377–388, 1989.

[Mau92] L. Maugis. Adequacy of active oodbms to flight data processing servers. Master's thesis, National School of Civil Aviation / University of Florida, E470-CSE, Gainesville, FL 32611, August 1992.

[Mck92] Bill Mckenna. Vocano query optimizer generator manual. Technical report, University of Colorado,Boulder, 1992.

[Min88] Jack Minker, editor. *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann Publishers,INC, 1988.

[MP90] C. B. Medeiros and P. Pfeffer. A Mechanism for Managing Rules in an Object-oriented Database. Technical report, GIP Altair, Sept. 1990.

[RC88] A. Rosenthal and U. Chakravarthy. Anatomy of a Modular Multiple Query Optimizer. In *Proceedings 14th International Conference on Very Large Data Bases*, pages 230–239, Los Angeles, CA, Sept. 1988.

[RCBB89] A. Rosenthal, U. S. Chakravarthy, B. Blaustein, and J. Blakeley. Situation Monitoring in Active Databases. In *Proc. of the 15th Int'l Conf. on Very Large Databases*, pages 455–464, Amsterdam, Aug. 1989.

[Sel86] T. Sellis. Global Query Optimization. In *Proceedings of SIGMOD*, pages 191–205, 1986.

[SHP87] M. Stonebraker, M. Hanson, and S. Potamianos. A Rule manager for Relational database Systems. Technical report, Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, 1987.

[SHP88] M. Stonebraker, M. Hanson, and S. Potamianos. The POSTGRES rule manager. *IEEE Transactions on Software Engineering*, 14(7):897–907, Jul. 1988.

[SKL89] S. Y. W. Su, V. Krishnamurthy, and H. Lam. "An Object-Oriented Semantic Association Model (OSAM*)". *Theoretical Issues and Applications in Industrial Engineering and Manufacturing*, pages 242–251, 1989.

[SR86] M. Stonebraker and L. Rowe. The Design of POSTGRES. In *Proceedings of ACM-SIGMOD*, pages 340–355, 1986.

[Sto90] M. Stonebraker. On rules,procedures,caching and views in database systems. In *proc. of the ACM SIGMOD*, May 1990.

[VCJ] Yannis Vassiliou, Jim Clifford, and Matthias Jarke. Data access requirements of knowledge -based systems.

[WBT92] D. Wells, J. A. Blakeley, and C. W. Thompson. Architecture of an open object-oriented database management system. *IEEE Computer*, 25(10), October 1992.

[WCB91] J. Widom, R. J. Cochrane, and . Lindsay B, G. Implemented Set-Oriented Production Rules as an Extension of Starburst. In *Proceedings 17th International Conference on Very Large Data Bases*, pages 275–286, Barcelona (Catalonia, Spain), Sep. 1991.

[WF90] J. Widom and S. Finkelstein. Set-Oriented Production Rules in Relational Database Systems. In *Proc. of ACM-SIGMOD*, pages 259–270, May 1990.