Johann Christoph Freytag, David Maier,
Gottfried Vossen (editors):

Query Processing in Object-Oriented, Complex-Object and Nested Relation Databases

Dagstuhl-Seminar-Report; 15
17.-21.6.1991 (9125)

ISSN 0940-1121

Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloß Dagstuhl
Informatik, Bau 36
Universität des Saarlandes
W - 6600 Saarbrücken
Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397
e-mail: dagstuhl@dag.uni-sb.de

Query Processing in Object-Oriented, Complex-Object and Nested Relation Databases

Organizers: J.C. Freytag, D. Maier, G. Vossen

1 Introduction

There is an abundance of new types of database system on the horizon: behavioral and structural object-oriented database systems, extended relational databases, multimedia databases and database systems based on semantic data models. These efforts have yielded more than just new data model or paper systems. All the types of systems mentioned are represented by working prototypes or even initial commercial offerings. The initial focus of these systems has been greater expressivity and supporting modes of access of new database applications, such as CAD and CASE. However, on the whole these new systems are weak in an area where current relational systems excel: efficient querying via a declarative language.

The need to provide such a capability in these new database systems is widely acknowledged, but the challenges are many:

1. The algebraic formalisms used to represent queries for optimization and evaluation need to be extended to handle new data model features such as added base types, ordered type constructors (e.g., lists and arrays), object identifiers and user-defined data types.

2. Relational query processing exploits the homogeneity of large collections of data to carry out set-at-a-time processing. While these new database systems intend to support large collections, heterogeneity creeps in from a variety of sources: different levels of nesting or repetition, multiple implementations for a type or a method, subtype hierarchies and union types to name a few. Can set-at-a-time processing be adapted to tolerate such variations?

3. Many of these systems provide for extensibility by the application programmer, database designer or DBMS implementor. Query optimizers must be constructed so that they can be extended in parallel with the models and access methods.

4. The encapsulation provided by class- or type-definition mechanisms found in some of these systems obscures the global knowledge required by a query optimizer to deduce alternative execution. Do there exist processing strategies that allow the system to manage these two conflicting concepts in such a way that they can coexist together? Based on requirements and/or user specifications one might give preference to maintain encapsulation or to provide

global knowledge for exhaustive optimization. Do there exist intermediate approaches between these two extremes (i.e., not giving up encapsulation completely thus not gaining complete global knowledge of the request) and if so what kind of optimization is possible in those?

5. To support efficient execution of user requests in object-oriented, complex object, or nested relation database systems, new kinds of access structures need to be offered and explored. Furthermore, relationships between objects on the logical level might impact the organization of objects on the storage level. Such physical organization then needs to be maintained over time. What kinds of concepts, data structures, and algorithms are necessary to achieve optimal storage and processing of objects?

6. Most relational optimization is done at compile time based on cost estimations that are derived by cost functions. The question is if such a "static" model is still adequate for models with much richer concepts than the relation model. Are there alternative ways of optimizing user requests, such as some kind of "run time" optimizations that exhibit a more "dynamic" behavior to take better into account the existing object organization and to adapt more easily to possible changes on various levels.

7. Most relational systems focus on physical optimization, i.e., taking into account the physical layout of data on disk possibly supported by auxiliary structures such as indexes. A semantically richer environment provides much more information that should be included in the processing of user requests, in particular information embedded in the semantic concepts offered by the various models. Taking into account such information leads to a kind of "logical" optimization that goes beyond current optimization in relational systems. The question arises what kind of optimizations are possible and what kind are desirable for these advanced database models.

8. All these new aspects raise the immediate question if the current architectures for query processing components in relational systems are adequate for also implementing such components for advanced database systems. The higher complexity in query processing might suggest alternative architectures that go beyond simply extending existing ones to the new requirements.

The intent of this workshop was to bring together academic and industrial researchers working in query processing to discuss their initial efforts in solving these problems, to set forth new problems they have encountered in the process, and to generate ideas for new query processing strategies in advanced data models.

The meeting was the first database workshop held at Dagstuhl, and brought together 32 scientists from 8 different countries (see Section 5). During the week,

22 presentations were given (see Sections 2 and 3) with plenty of discussion time after each; in addition, two afternoons were reserved for discussion of special topics in small groups (see Section 4). The schedule for the different sessions and their contents was not determined in advance, rather the organizers wanted to be flexible in this matter and to react dynamically to requests and the questions raised by all partcipants during the various session. The special topics that we discussed in small groups were collected during the presentation sessions from the audience. In this way we were able to cover additional aspects of query processing that were not addressed by the presentations.

We felt that all participants enjoyed the workshop, and we wish to thank the Dagstuhl staff for ensuring that everything ran so smoothly.

2 Final Program

Monday, June 17, 1991

Opening Remarks, Introductory Statement by each Participant 9.00 – 10.00 h
Chair: G. Vossen, FRG

Session 1: Indexing 10.30 – 12.30 h
Chair: J.C. Freytag, FRG

Indexing Techniques for Object-Oriented Databases
E. Bertino, Italy

Query Processing in GOM
G. Moerkotte, FRG

Session 2: Optimization 2.30 – 5.45 h
Chair: D. Maier, USA

Evaluation and Optimization of Complex Object Selections
J. van den Bussche, Belgium

Towards a Unification of Rewrite Based Optimization Techniques for Object-Oriented Queries
S. Cluet, France

Optimization of Complex-Object Queries in PRIMA — Statement of Problems
H. Schöning, FRG

Open Discussion; Proposals for Special Interest Groups

Tuesday, June 18, 1991

Session 3: Authorization and Methods Processing 9.00 – 12.15 h
Chair: H.-J. Schek, Switzerland

The Revelation Project: Query Processing in Object-Oriented Databases
D. Maier, USA

Implementation of the Object-Oriented Data Model TM

P. Apers, The Netherlands

Supporting Access Control in an Object-Oriented Database Language
P. Lyngbaek, USA

Session 4: Models 2.30 – 6.15 h
Chair: G. Lausen, FRG

What Results and Models Would Speed Progress for Other Researchers
A. Rosenthal, USA

Special Interest Groups
Open Discussion

Wednesday, June 19, 1991

Session 5: Complex Objects/Nested Relations 9 – 12.30 h
Chair: K. Kulkarni, USA

Design and Implementation of SQL/XNF in Starburst's Extensible Database System
B. Mitschang, FRG

Query Optimization in COOL — an Object-Oriented Database Query Language Based on Nested Relations
M. Scholl, Switzerland

Joint Presentation:
Research in Query Processing at IFI/University of Zurich
A. Geppert, Switzerland
Optimization of Object-Oriented Queries Based on the NO²-Algebra
B. Demuth, FRG
Overview of the Ithaca Project
Th. Gorchs, FRG

Thursday, June 21, 1991

Session 6: Parallelism 9.00 – 12.15 h
Chair: G. Graefe, USA

Parallel Processing of Complex Objects over a Storage Hierarchy
Y. Kornatzky, Israel

Dynamic Parallel Query Processing
M. Kersten, The Netherlands

Query Processing in Volcano
G. Graefe, USA

Session 7: Architecture 2.00 – 6.00 h
Chair: P. Lyngbaek, USA

Query Processing in Client-Server Systems with Object-Orientation
P. Dadam, FRG

Evaluating Complex Queries Accessing Hierarchically Structured Objects by Using Path Indexes
U. Kessler, FRG

Special Interest Groups
Open Discussion

Friday, June 22, 1991

Session 8: Deductive Approaches 9.00 – 11.00 h
Chair: J.C. Freytag, FRG

Optimizing Queries with Structural Axioms of an Object-Centered Data Model
M. Jeusfeld, FRG

An Object-Oriented View on a Deductive Database
G. Lausen, FRG

Session 9: Languages 11.15 – 13.00 h
Chair: J.C. Freytag, FRG

Tagging as an Alternative to Creating New Objects

M. Gyssens, Belgium

ADT-based Type Systems for SQL
K. Kulkarni, USA

Session 10: Final Discussion 2.00 – 4.00 h

3 Abstracts of Presentations

The following abstracts of presentations appear in alphabetical order of speakers.

Implementation of the Object-Oriented Data Model TM

Peter M.G. Apers, University of Twente, The Netherlands

TM is an object-oriented data model currently being developed at the University of Twente. Its main characteristics are:

- high abstraction level for easy specification of attributes, methods, and constraints;
- type constructors for complex objects are record, set, list, and variant;
- inheritance of attributes, methods, and constraints;
- formal semantics based on type and set theory, lambda calculus, and first order logic.

The language is currently being tested in a geographic and a complex hospital application.

ADL is an algebra for complex objects with the same type constructors as TM. It has higher order functions such as *MAP, SEL*, and *GEN*, besides the normal operations such as *PROD, NEST, UNNEST* etc. ADL allows for easy rewriting of expressions to obtain a more efficient execution plan for queries.

A translation of TM types and expressions to ADL is given. This translation also takes inheritance into account. Furthermore, optimization rules in ADL are discussed.

Future work concerns logical query language DTL for TM and an implementation of both of them.

Indexing Techniques for Object-Oriented Databases

Elisa Bertino, University of Genova, Italy

In this work several issues concerning index organizations for object-oriented databases are discussed and proposed techniques are surveyed. For the purpose of the discussion, an object-oriented database is considered organized into two orthogonal dimensions: aggregation graphs, and inheritance hierarchies. First, indexing techniques for efficiently traversing aggregation graphs are surveyed. The basic idea is to maintain into separate structures object references along aggregation branches that are frequently traversed in queries. Techniques surveyed include: multi-index,

nested index, path index, join index, access relation. Then indexing techniques for inheritance hierarchies are discussed and two different organizations are presented: single-class index, and class hierarchy index. In the first organization, an index is maintained for each class in the hierarchy. In the second, the index is stored by all classes in the hierarchy. In addition, integrated organizations are presented, that support indexing along both inheritance and aggregation graphs. Finally, the problem of method caching and precomputation is briefly discussed.

Evaluation and Optimization of Complex Object Selections

Jan van den Bussche, University of Antwerp, Belgium

We provide a general framework for declarative selection operations for complex object databases, based on the safe calculus for complex objects. Within this framework, we consider a class of "single pass-evaluable" selection operations. We show how such selection operations can be succinctly expressed by programs that use only very simple positive existential selections. Also, a syntactic criterion is developed for the *commutation* of two such positive existential selections. These two results are then jointly applied to the problem of *optimizing* complex object selections, which is much more complicated than in classical relational databases. Our results also find an application in rule systems for complex objects.

Towards a Unification of Rewrite Based Optimization Techniques for Object-Oriented Queries

Sophie Cluet, Altair, France

We present a formalism for the logical layer of an object-oriented query optimizer that subsumes two well-known optimization approaches: the Orion technique based on classes extensions and the algebra based query rewritings. The formalism also allows easy and exhaustive factorization of common query subexpressions. Furthermore, it uses information on object placement policies and indices to limit the search space for an equivalent expression, thereby reducing the rewriting phase.

Query Processing in Client-Server Systems with Object Orientation

Peter Dadam, University of Ulm, FRG

The availability of fast and cheap microprocessor technology is currently leading to a quick growth of PC's and workstations in industry and other organizations. In business administration oriented application areas the trend is to "decentralize" some of the applications by off-loading some of the application programs from the host systems down to PC's. As these PC's usually need access to data residing in host-based or centralized database systems this trend is leading quite naturally to client-server solutions in the one way or another. Opposed to that, engineering tasks have been usually performed on specialized (isolated) systems rather than on central host systems already in the past. The quick growth of workstation installations in conjunction with the trend towards Computer Integrated Manufacturing (CIM), however, makes some kind of global control and integration via a (logically) centralized engineering database more and more important. Client-server architectures are the natural choice here too. Unfortunately, the DBMS technology currently available is not very well suited for this task. While the functionality and expressive power of SQL only shows some weaknesses in business administration oriented applications, it is practically not usable for engineering applications.

The weakness of relational database technology to handle complex structured data objects is known and has led to many research and development efforts under the label "object-oriented DBMSs". One can distinguish two directions of research and development in this area: One is heavily inspired by object-oriented programming languages like C++ or Smalltalk while the other is more in the tradition of relational database systems. Unfortunately, both approaches often seem to address "different worlds". The talk tries to outline that some of the "big" differences may not be that big as they look like at first glance. It also is a pleading for pushing towards a common view on data and objects rather than pushing for very different types of systems which would make integrated solutions as required for CIM, for example, very hard to achieve.

Formalization and Optimization of Queries in the NooDLE Database System

Birgit Demuth, Technical University of Dresden, FRG
Andreas Geppert, University of Zurich, Switzerland
Thorsten Gorchs, Siemens-Nixdorf, FRG

NooDLE is the database system of the ITHACA (Integrated Toolkit for Highly Advanced Computer Applications, an ESPRIT-II project) software production environment. NO^2 (New Object-Oriented Data Model), the data model of NooDLE

is a structurally object-oriented data model. NO^2 objects are pairs (OID, value) where value constructors (set, tuple, list, and array) can be combined in a completely orthogonal manner. Object types have assigned a value set (describing the permitted values for instances of the type) and an extension to them. Furthermore, NO^2 supports type hierarchies (multiple inheritance). Permitted object structures are subobject and general references (comparable to ORION).

Declarative or ad-hoc access is provided by the Quod query language (Querying Object-Oriented Databases). Among other features, Quod supports recursive queries. Furthermore, access to NooDLE databases is provided via the programming languages C++ and Cool (Combined Object-Oriented Language). Thus, integration of NOODLE and (say) Cool results in full object-orientation.

A formalization of NO^2 and Quod is given by the NO^2 algebra. Each value set (and each extension) corresponds to one carrier of the many-sorted algebra. Supported operators are constructors, projection (for objects, lists, tuples, and arrays), union and difference, flatten, image, and (last not least) selection for sets and lists. Furthermore, joins can be expressed by other operations, but may be introduced for the sake of optimization.

Algebraic optimization takes idempotence, commutativity, associativity, and distributivity properties of operations into account. Some rules (based on these properties) carry over from the relational algebra. Other rules are related to the specific features of NO^2; thus we specified inheritance, subobject, and navigational rules. This work was partially supported by ESPRIT-II (Demuth, Gorchs) and KWF, Switzerland (Geppert).

Query Processing in Volcano

Goetz Graefe, University of Colorado, USA

Volcano is an extensible and parallel query processing system. Its optimizer generator creates optimizer source code from a data model description file similarly to the EXODUS optimizer generator, but significantly improves the search strategy and its support for physical properties like sort order and data distribution in parallel systems. The execution engine consists of an extensible set of operators realizing mechanisms for query execution such that policies (strategies) can be determined by a query optimizer or a human experimenter.

Its novel operators are the two "meta-operators": the "choose-plan" operator for dynamic plans and the "exchange" operator that encapsulates all issues of parallel execution, including process creation, scheduling, flow control, data transfer. It even hides the underlying hardware architecture from the "work" operators which can be implemented in a sequential environment but parallelized by combining them with exchange operators.

Tagging as an Alternative to Creating New Objects

Marc Gyssens, University of Limburg, Belgium

Based on the observation that graphs play an important role in the representation of databases, an algebra is presented for the manipulation of binary relations, i.e., of directed unlabeled graphs. The algebra is based on early work by Tarski. The key notion that is added is tagging which is needed for giving the model both enough modeling power and enough querying power. Tagging can also be seen as a value-based counterpart to object creation in object-oriented data models. In this work, we present a general formal framework for tagging that incorporates several variations of tagging as a special case. We also show that, in some sense, tagging can be seen as a generic operation. It also follows that adding tagging to the Tarski algebra together with a while-construct results in a computationally complete database language. Finally we show how tagging can be used to represent sets, especially in simulating the nest operator of the nested relational algebra.

This work was done together with L. Saxton, University of Regina, Sask., Canada, and D. van Gucht, Indiana University, Bloomington, IN, USA.

Optimizing Queries with Structural Axioms of an Object-Centered Data Model

Manfred Jeusfeld, Universität Passau, FRG

A key technology that made relational databases a success is the efficient evaluation of declarative expressions, in their incarnations as queries, deductive rules, and integrity constraints. Surprisingly, these efficient evaluation algorithms can be transferred easily to an object-centered data model, and take advantage of the additional structure in such an environment.

Rather than over relations or domains variables in a query range over classes with finitely many instances:

$$\forall\, x_1/c_1, \ldots, x_n/c_n\ \Phi \Rightarrow \mathit{Answer}(x_1, \ldots, x_n)$$

which is a shorthand for

$$\forall\, x_1, \ldots, x_n\ \mathit{In}(x_1, c_1) \wedge \ldots \wedge \mathit{In}(x_n, c_n) \wedge \Phi \Rightarrow \mathit{Answer}(x_1, \ldots, x_n)$$

The formula Φ makes a statement about relationships between objects (here we uniformly regard values and classes as objects). This is expressed by a literal $A(x, l, y)$ which is sometimes written as $x.l = y$. All variables are assigned to classes. Consequently, it is possible to determine the attribute definition (c, l, d) at the class level that corresponds to a literal occurrence $A(x, l, y)$ in a query.

A usual constraint in object-oriented languages demands the attributes of an instance of a class to be instances of the attribute definitions. Thus, assigning $A(x, l, y)$ to the attribute definition (c, l, d) guarantees that $In(x, c)$ and $In(y, d)$ hold. Therefore, each occurrence of such literals can be eliminated if they appear in conjunction with $A(x, l, y)$.

One may argue that a non-typed query language wouldn't have caused the problem with the In-literals at all. However, in that case the concerned attribute definition (c, l, d) is no longer unique. Since the same attribute label l can be used for different classes which yields a bigger search space.

This optimization technique has been applied also to deductive rules and integrity constraints. Implementation was done within the object base ConceptBase.

Dynamic Parallel Query Processing

Martin Kersten, CWI, Amsterdam, The Netherlands

Traditional query optimizers for multiprocessor database systems produce a mostly fixed query evaluation plan based on assumptions about data distribution and processor workloads. However, these assumptions may not hold, at query execution time due to contention caused by concurrent use of the system or lack of precision in the derivation of the query plan. In this task, we propose a dynamic query processing scheme based on subdividing the query into subtasks and scheduling these adequately at runtime. We present the results obtained by simulation of a queueing network model of the proposed software architecture.

This work was done together with C.A. van den Berg and S. Shair-Ali, CWI, Amsterdam.

Evaluating Complex Queries Accessing Hierarchically Structures Objects by Using Path Indexes

Ulrich Kessler, University of Ulm, FRG

To be generally applicable, an object-oriented database management system supporting complex objects should not only provide the retrieval of complex objects as a whole but also of arbitrary subparts. For that reason, many systems offer a descriptive query language to express arbitrary selections and projections (sometimes called multi-target queries). Typically, complex queries are made up of subqueries. Each of it refers to a multi-valued attribute of the object for which a selection or projection shall be applied. In this context we can discriminate between independent (sub)queries which are accessing "top-level" objects and dependent subqueries which are selecting subobjects belonging to multi-valued attributes of parent objects. They are called dependent queries because they are only fully specified after

the parent object owning the multi-valued attribute has been selected. To execute such complex queries efficiently the database management system should be able to to optimize them and to chose secondary access paths like Indexes, for example, automatically as it is done today in relational systems. Within the context we are considering, the queries to be analyzed and optimized can easily become rather complex; much more complex than typical queries in relational systems. Therefore, performing query optimization efficiently becomes an important issue by itself. Our approach is to divide the queries into the subqueries they are composed of and to decide on the evaluation strategy for each of these subqueries almost independently. For that reason, we will discuss in the second part of the talk alternative strategies to evaluate subqueries without and with using indexes. As many systems are internally using hierarchical data structures to store complex objects we will assume path indexes because they are well suited to invert such data structures.

When evaluating a dependent query it has to be guaranteed that only subobjects belonging to the parent object under consideration will be returned. One way to ensure this restriction is to use not only a value-based selection predicate as a search predicate within an Index structure but in addition also a so-called address predicate containing the identifier of the parent object. By doing so the index manager will read and return only identifiers of subobjects belonging to the respective parent object. An alternative method will be to retrieve in a first step all index entries satisfying the value-based predicate from the index structure and store them in an intermediate result table. Because in this case the address predicate is not evaluated, the result set will contain identifiers of all matching subobjects of all parent objects. Therefore, whenever evaluating the related query for an actual parent object an additional associative search will be performed within this intermediate result to identify the set of identifiers of subobjects belonging to the actual parent object. Both evaluation methods for dependent queries are useful under different circumstances. If these methods are combined with methods to evaluate independent queries as for example a complete class scan or an index scan even complex queries can be optimized.

Parallel Processing of Complex Objects over a Storage Hierarchy

Yoram Kornatzky, Hebrew University, Israel

We consider processing of complex objects by a shared-nothing multiprocessor architecture whose storage system consists of multiple levels. Complex objects may be spread over the different levels of the storage hierarchy, where in particular long fields are stored in lower levels. Our goal is to develop a realistic theory of data movement between levels, which will form the basis for query processing and optimization. We present a simple visual model for data organization which identifies the basic tradeoffs in processing queries over the storage hierarchy.

For an ideal model of queries processing complex objects in ascending order of components' storage level, we propose an elevator algorithm analogous to the familiar elevator disk scheduling algorithm. The algorithm permits maximal bundling of accesses by different queries to the same storage level. To accommodate non-ideal queries we suggest a new temporal clustering concept consisting of grouping sub-queries accessing common objects and executing them in parallel.

An increasing number of future applications will be written in database programming languages. In the discussed environment, physical data independence of programs mandates the use of powerful program compilers, parallelizers, and optimizers. We provide a foundation for compile-time scheduling of such programs by developing a suitable notion of iteration space. Based on the iteration space description of programs we suggest a compilation scheme into queries processed by the elevator algorithm.

ADT-Based Type Systems for SQL

Krishna Kulkarni, DEC, USA

Currently, most database vendors support the Structured Query Language (SQL), relational query language adopted by both ANSI and ISO as the database language standard. An examination of the type system associated with the SQL language reveals many serious shortcomings. This talk reports on an effort to add an extensible type system to SQL that is based on the notion of abstract data types (ADTs) and query language extensions that deal with the new type system while maintaining upward compatibility. The types in this system are abstract (obey strict encapsulation) and orthogonal (allow arbitrarily complex types to be built). In addition, the type system supports both value and object semantics, and supports the usual notions of inheritance, polymorphic functions and dynamic binding. The talk also covers the initial change proposals that have been submitted to both ANSI and ISO for incorporation into SQL3.

This work was done jointly with with Umesh Dayal, Jim Melton, Jonathan Bauer and Mike Kelley, all from Digital Equipment.

An Object-Oriented View on a Deductive Database

Georg Lausen, University of Mannheim, FRG

A subset of the F-logic language is considered which can be encoded in first-order logic. For the encoding Datalog with function symbols is sufficient. To make query evaluation more efficient, a parallelization is discussed. The proposed techniques generalize previous work by Wolfson et al. and are based on rewriting the original program. These techniques seem to be promising for the implementation of a rule-based, object-oriented language on top of a deductive database.

Supporting Access Control in an Object-Oriented Database Language

Peter Lyngbaek, HP Labs, USA

An important functionality of a DBMS is its support of access control. Most relational DBMSs have security subsystems that support some form of discretionary access control. However, little work has been reported on authorization features in database systems based on newer database models, e.g. semantic, functional, logic, and object-oriented models. Such high-level models may support more flexible authorization and finer levels of access control than the relational model. This talk presents an approach for providing access control in QSQL, an object-oriented database language that supports, among others, user-defined abstract data types, multiple inheritance and late binding. The authorization scheme is based on a single concept: that of controlling function evaluation. The talk discusses how the authorization model can be realized using existing QSQL mechanisms, i.e. subtyping, user-defined operations, and function resolution. It also presents two novel constructs: guard functions and proxy functions, which are useful in providing database security in a flexible and non-invasive manner. Various issues related to the language semantics are examined.

This work was done together with R. Ahad and E. Onuegbe, both Hewlett-Packard Company.

The Revelation Project: Query Processing in Object-Oriented Databases

David Maier, Oregon Graduate Institute

There is an abundance of new types of database system on the horizon: behavioral and structural object-oriented database systems, extended relational databases, multimedia databases and database systems based on semantic data models. These efforts have yielded more than just new data models or paper systems. All the types of systems mentioned are represented by working prototypes or even initial commercial offerings. The initial focus of these systems has been greater expressivity and supporting modes of access of new database applications, such as CAD and CASE. However, on the whole these new systems are weak in an area where current relational systems excel: efficient querying via a declarative language.

Relational query processing exploits the homogeneity of large collections of data to carry out set-at-a-time processing. While these new database systems intend to support large collections, heterogeneity creeps in from a variety of sources: different levels of nesting or repetition, multiple implementations for a type or a method, subtype hierarchies and union types to name a few. There are also new data model features to consider: added base types, ordered type constructors (e.g., lists and arrays), object identifiers and encapsulation. The Revelation project being conducted at OGI and University of Colorado at Boulder seeks to extend set-at-a-time processing to handle such variations.

This talk explains the goals, top-level architecture and initial results of the Revelation project. One main topic will be the design of the type definition and implementation description portions of the data model, and how they interact with the "revealer", a trusted system component that is allowed to break the encapsulation of abstract data types. Another main topic will be an "assembly" operator added to the Volcano query evaluation system to support access to complex objects.

Design and Implementation of SQL/XNF in Starburst's Extensible Database System

Bernhard Mitschang, University of Kaiserslautern, FRG

Complex applications, such as design applications, multi-media and AI applications, and even enhanced business applications can benefit significantly from a database language that supports complex objects. The data used by such applications are often shared with more traditional applications, such as cost accounting, project management, etc. Hence, sharing of the data among traditional applications and complex object applications is important.

Our approach, called SQL Extended Normal Form (short SQL/XNF), enhances the relational language SQL towards a complex object concept that supports

Entity-Relationship (E-R) model as well as Object Oriented concepts. The language allows sharing of the database among normal form SQL applications and complex object applications. SQL/XNF provides sub-object sharing and recursion, all based on its powerful complex object constructor concept, which is closed under the language operations. SDQL/XNF DDL and DML are superset of SQL, and are downward compatible with SQL.

In this talk we concentrate on the main ideas underlying intergration of SQL/XNF into the Starburst DBMS. We present extensions to the catalog and QGM structures for XNF. We discuss the semantic routines used to generate such structures from XNF queries. At the end we present the translation/optimization algorithms for conversion of XNF QGM to NF QGM. Enough background information on internals of Starburst, including QGM, will be provided to make the talk useful for general audience.

This work was done together with H. Pirahesh at the IBM Almaden Research Center while the author was on leave from University of Kaiserslautern to IBM Almaden Research Center, San Jose, California.

Query Processing in GOM

Guido Moerkotte, Universität Karlsruhe, FRG

Object-oriented database systems are emerging as the next generation databases for non-standard applications, e.g., VLSI-design, mechanical CAD/CAM, software engineering, etc. While the large body of knowledge of relational query optimization techniques can be utilized as a starting point for object-oriented query optimization the full exploitation of the object-oriented paradigm requires new, customized optimization techniques—not merely the assimilation of relational methods. This talk describes such an optimization strategy used in the GOM project which combines established relational methods with new techniques designed for object models. The optimization method unites two concepts: (1) *access support relations* and (2) *rule-based query optimization*. Access support relations constitute an index structure that is tailored for accessing objects along reference chains leading from one object to another via single-valued or set-valued attributes. The idea is to redundantly maintain frequently traversed reference chains separate from the object representation. The rule-based query optimizer generates for a declaratively stated query an evaluation plan that utilizes as much as possible the existing access support relations. This makes the exploitation of access support relations entirely transparent to the database user. The rule-based query optimizer is particularly amenable to incorporating search heuristics in order to prune the search space for an optimal (or near-optimal) query evaluation plan.

What Results and Models Would Speed Progress for other Researchers

Arnon Rosenthal, Xerox AIT, USA

We first describe ambitious problems that could serve as goals to motivate architectures and research plans: Optimization for OODB programming languages, and extensible optimization for OODBMS query languages. For DBPL's, our goal is to provide adequate performance for high-level expressions that ignore issues such as representations. For extensibility, we emphasize building optimizers that have a controller which coordinates expertise (collections of transformations) obtained from diverse sources. We discuss how theory might be used to simplify parts of complex optimizers, making it more flexible to add new functionality; we also discuss the need for these formalizations to represent the largest possible fraction of the query compilation process. Finally, we discuss challenges added by object-orientation, such as multiple levels of abstraction, cost models that are undefined for certain operators, and a "projection server" that models access to parts of data clusters.

Optimization of Complex-Object Queries in PRIMA — Statement of Problems

Harald Schöning, University of Kaiserslautern, FRG

The MAD (molecule-atom data) model allows the dynamic construction of complex objects via implicit joins (using an identifier/reference concept). Binary relationships among basic objects (called atoms, which correspond to tuples in the relational model) are represented symmetrically by a pair of reference attributes. The MAD model allows network-like and recursive object structures, which are formed from hierarchical ones by specialized operators. The basic operator is *Construction of simple molecules*, which builds up a set of hierarchical molecules and performs selections that can be evaluated on a single molecule (i.e. no query nesting). It is also able to perform projections on atoms and attributes. All other operators get their input from this operator in a pipelined way. Often, however, Construction of simple molecules is the only operator representing a query, since in many cases the complex object facilities are sufficient to model the application's objects without explicit join ro other higher operators. Therefore, it is necessary to direct one's attention to the optimization of the execution of Construction of simple molecules.

Some optimization problems occurring in connection with this operator are identified:

- The classical join order and join method problems do not apply to the MAD model; here we have to solve the hierarchical join schedule problem.

- This problem covers the question of entry point selection and search strategies as well as the amount of parallelism to be exploited within each hierarchical join.
- Join sequences can be replaced by cluster accesses. This must be covered by the plan generation within the optimizer. On the other hand, the system should propose clustering structures which are useful for as many queries as possible.
- It is not clear, whether every possible kind of parallelism really should be exploited, when the amount of parallelism supported by the hardware can be reached without doing so.
- The kind of information to be kept in the system statistics must be determined.
- A language for the formulation of optimization rules has to be developed.

Query Optimization in COOL — An Object-Oriented Database Query Language Based on Nested Relations

Marc Scholl, ETH Zurich, Switzerland

COOL (Complex-Object-Oriented Language) is the query language of the object-oriented database system currently being developed at ETH Zurich. It is "based on nested relations" in two ways: (1) it can be seen as an extension of the nested relational algebra that has been developed within the DASDBS project, and (2) the target system for the implementation is the DASDBS nested relational storage manager.

The talk gives a short overview of the data model, particularly we emphasize object-preserving query semantics (that is, the query results are sets of existing objects) and the separation between the notions of a type (a set of functions that can be applied to the type's instances) and that of a class (an object representing a collection of objects that are instances of the type associated with the class).

The second part of the talk illustrates the various choices we have in representing the physical database layout for a given COOL schema in the form of nested relations. The basic options provided include (1) objects for object-valued functions, (2) references versus objects for object valued functions, (3) with versus without backward references, and (4) with versus without physical references (TIDs) in addition to logical references (OIDs). We then report on a first prototype implementation of a "physical DB design expert system" exploiting some (but not yet all) of the options. The systems takes as input a COOL schema, a description

of the anticipated (or observed) transaction load, and information on the cardinality and size of the DB objects. The output is a set of proposed physical designs based on cost estimates for the trans- action mix obtained from a cost model.

Finally, we present two approaches to the query optimization problem arising from the flexible storage structures: depending on the design optimizer's (or human DBA's) choice, different internal (nested relational) query formulations have to be generated for the same COOL query against the logical COOL schema. The first approach is an algebraic query rewrite one: COOL queries are first mapped to a (hypothetical) nested relation that corresponds to the default physical design. In particular, this query will contain joins (over OID attributes) for all the (object-valued) function applications (traversals between object types). The challenge then is to eliminate those joins, that have internally been materialized in a hierarchical cluster. The second approach uses the COOL query graph, where each edge represents a function application. Now we start marking the query graph by attaching labels to the edges indicating whether they are supported by either of: references, materialization, "join indices", or not at all (which means it has to be executed as a value-based join).

4 Specific Discussion Topics

The organizers considered it important to get discussions on specific topics in the area of query processing started early on, and to provide enough opportunities to keep them going. Two things were done in this respect: First, every participant was asked to answer the following question at the beginning of the first morning:

> *If there was one question you would like to have answered during this week, what would it be?*

Second, on Tuesday and Thursday afternoon, several small groups met for discussing a variety of special query processing topics, which will be listed below together with a brief statement of their results.

The questions that were raised by participants at the beginning of the workshop were the following (in the order they were posed):

1. What is *conceptually* new in query processing and optimization in object-oriented, complex-object, and nested relational databases? (Vossen)
2. What is a general mechanism for optimizing queries on ADTs? (Kulkarni)
3. What does it mean in a CAD or CIM context to have complex queries and how can database query processing help to optimize and evaluate them? (Freytag)
4. Is there a convergence of the nested relational, complex object, and object-oriented directions? (Freytag)
5. Will there be a common foundation for query processing in new systems? (Lyngbaek)
6. Is it useful to work at an abstract level, or can we do everything in a flat implementation? (van den Bussche)
7. What new insights are possible into the problem of duplicate elimination? (Gyssens)
8. How to compile and efficiently implement database programming languages? (Kornatzky)
9. Can there be a good benchmark for a query optimizer? (Cluet)
10. Must an optimizer be messy or clean? (Cluet)

11. Is there a general framework connecting the approaches to query processing in deductive, nested relational, and complex object databases, and what degrees of freedom are needed in each? (Mitschang)

12. How to do query processing on bulk data types that are not sets? (Maier)

13. Why just look at object-orientation, instead of looking at query optimization in a multi-paradigm (object-oriented, logic-oriented, functional) context? (Apers)

14. Does traditional query processing technology work in a parallel environment? (Kersten)

15. What are the limits on query processing techniques that rely on term rewriting? (Kersten)

16. How to do query optimization in logic-oriented/object-oriented database systems? (Bertino)

17. What are good ideas to generalize known processing approaches to queries with nesting, subobjects, type hierarchies, behavior, recursion? (Demuth)

18. How to implement an object-oriented query optimizer on a database kernel? (Gorchs)

19. Is there a unified framework for query languages, especially algebraic ones? (Geppert)

20. How to use indexes to access complex data, and which cost formulas are appropriate for selecting query plans? (Kessler)

21. Is there an easy connection between deductive and object-oriented query processing? (Jaeger)

22. Will object-oriented database systems be simpler or more complicated than relational systems? (Jeusfeld)

23. Are these powerful languages really needed and, if so, by what applications? (Schöning)

24. How to implement complex object database systems with reasonable performance, how to index, manage main memory, put them on disk? (Ley)

25. Which new approaches can be obtained for efficient query processing of nested relations? (Rich)

26. What is the right level of generality in algebraic approaches that gets performance? (Graefe)

27. What should a query language for object-oriented database look like? (Moerkotte)

28. How do top-down techniques relate to bottom-up ones in parallelization of deductive languages? (Lausen)

29. How can algebraic optimization be extended, how can indexes, link fields be included, how can their elimination as being redundant be avoided? (Schek)

30. How to organize an optimizer in a heterogeneous environment? (Rosenthal)

31. How can we build real systems instead of leading isolated discussions? (Dadam)

The Tuesday and Thursday discussion groups came to conclusions on the following topics:

1. Views and derived information:
   Views are useful and necessary; they should be definable via queries. Hence it is important to look at the underlying language first, which should be capable of returning objects; as a consequence, views will be updatable. Related aspects include object identifier generation in rule-based languages, which consider views as derived classes.

2. Selectivity and cost models:
   In the best case, estimations will be *robust*, a requirement that is not unreasonable. Something like this would typically be based on estimating transitive closure computations. *Dynamic* plans might be a feasible solution.

3. Result formatting and distribution:
   A better understanding is needed of how the programming language in which a request is formulated uses the data that is returned; for example, that data maybe a byte-stream-like video data.

4. Imperative vs. logical vs. functional languages:
   The motivation for having languages that are complete with respect to all these paradigms seems doubtful; keeping query languages and programming languages separate is reasonable in many applications. A study of functional languages in the context of new database systems makes sense; studies need also be conducted on uniform vs. multilingual environments.

5. Optimizer organization:
   Rules are good building blocks for an optimizer, in particular rules which are

based on notions of equivalence and/or rules for refinement, but control is also needed (and more difficult). Not yet well understood is the influence of low-level optimization on high-level optimization. Also, organizing an optimizer as a multiple-inheritance hierarchy seems reasonable (where an optimizer is considered as an object receiving a query and returning an optimized one), but that alone will not be good enough.

6. Physical representations suited for OODBs:
   Pros and cons can be found for (nested) relational tuples and object-oriented pointers. Other structures to be considered include main-memory graph structures and "optical object-oriented storage systems" ("O_3S_2"), which might be available in the near future.

7. Handling of bulk types other than sets:
   Important aspects seem to be order and topology. An adequate notation is needed for corresponding query languages.

8. How many different algebras are needed:
   Three different classes exist: flat, nested, and object algebras. These have some transformation rules in common, but differ considerably in their rewriting strategies; a goal is to design the algebra which is powerful enough to express complex execution plans and that rules can be be derived for improving transformations.

9. Complex object assembly and application program interfaces:
   Assembly of objects should either exist as an operator or be based on "pulling together" relevant parts. The main question for APIs is to what extent a DBMS should have to rework data so that an application can easily use it; current approaches include precompilers and call-interfaces. Experience with real applications is needed.

5 List of Participants

1. Peter M.G. Apers
   University of Twente
   Computer Science Department
   Postbus 217
   NL-7500 AE Enschede, The Netherlands
   apers@cs.utwente.nl

2. Elisa Bertino
   Dipartimento di Matematica
   Universita di Genova
   Via L.B. Alberti, 4
   I-16132 Genova, Italy
   bertino@igecuniv.bitnet

3. Jan van den Bussche
   Dept. of Computer Science
   University of Antwerp
   Universiteitsplein 1
   B-2610 Wilrijk, Belgium
   vdbuss@ccu.uia.ac.be

4. Sophie Cluet
   GIP Altair
   Domain de Voluceau — Rocquencourt
   B.P. 105
   F-78153 Le Chesnay Cedex, France
   cluet@bdblues.altair.fr

5. Peter Dadam
   Universität Ulm
   Fakultät fur Informatik
   Oberer Eselsberg
   W-7900 Ulm, FRG
   dadam@rz.uni-ulm.dbp.de

6. Birgit Demuth
   Technische Universität Dresden
   Fakultät Informatik
   Institut für Datenbanken und KI
   Mommsenstraße 13
   O-8027 Dresden, FRG

7. Johann Christoph Freytag
   Database Systems Research Group/Munich
   Digital Equipment GmbH
   Rosenheimerstr. 116b
   W-8000 München 80, FRG
   freytag@dunant.enet.dec.com

8. Andreas Geppert
   Institut für Informatik
   Universität Zürich
   Winterthurerstr. 190
   CH-8057 Zürich, Switzerland

9. Thorsten Gorchs
   Siemens-Nixdorf
   Microprocessor Engineering GmbH
   AP 44
   Gustav-Meyer-Allee 1
   W-1000 Berlin 65, FRG
   gorchs.bln@sni.de

10. Götz Graefe
    University of Colorado
    Dept. of Computer Science
    Boulder, CO 80309-0430, USA
    graefe@cs.Colorado.EDU

11. Margret Gross-Hardt
    Institut für Informatik
    Universität Koblenz-Landau
    Rheinau 3-4
    W-5400 Koblenz, FRG

12. Marc Gyssens
    Dept. of Computer Science
    University of Limburg
    B-3610 Diepenbeek, Belgium
    gyssens@bdiluc01.bitnet

13. Ulrike Jaeger
    FORWISS Institut
    Orleansstr. 34
    W-8000 München 80, FRG
    jaeger@forwiss.tu-muenchen.de

14. Manfred Jeusfeld
Universität Passau
Lehrstuhl für Dialogorientierte Systeme
Postfach 2540
W-8390 Passau, FRG
jeusfeld@andorfer.fmi.uni-passau.de

15. Martin Kersten
CWI
Kruislaan 413
NL-1098 SJ Amsterdam, The Netherlands
mk@cwi.nl

16. Ullrich Kessler
Universität Ulm
Abt. Betriebliche Informationssysteme/CIM
Oberer Eselsberg
W-7900 Ulm, FRG
kess_ull@rz.uni-ulm.dbp.de

17. Yoram Kornatzky
Dept. of Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
yoramk@cs.huji.ac.il

18. Krishna Kulkarni
Database Systems Research
Digital Equipment Corp. CXN1/2
1175 Chapel Hills Drive
Colorado Springs, CO 80920, USA
kulkarni@cookie.dec.com

19. Georg Lausen
Praktische Informatik III
Universität Mannheim
Seminargebäude A5
W-6800 Mannheim, FRG
lausen@pi3.informatik.uni-mannheim.dbp.de

20. Michael Ley
Fachbereich 4 — Informatik
Universität Trier
Postfach 3825
W-5500 Trier, FRG
ley@uni-trier.dbp.de

21. Peter Lyngbaek
HP Labs
DB Technology Department
1501 Page Mill Road
Palo Alto, CA 94304-1181, USA
lyngbaek@hplabs.hp.COM

22. David Maier
Oregon Graduate Institute
Department of Computer Science
19600 von Neumann Dr.
Beaverton, OR 97006, USA
maier@cse.ogi.edu

23. Ute Masermann
Institut für Informatik
Universität Koblenz-Landau
Rheinau 3-4
W-5400 Koblenz, FRG
maserman@degas.uni-koblenz.de

24. Bernhard Mitschang
Universität Kaiserslautern
Fachbereich Informatik
Erwin-Schrödinger-Str.
W-6750 Kaiserslautern, FRG
mitsch@informatik.uni-kl.de

25. Guido Moerkotte
IPD
Fakultät für Informatik
Universität Karlsruhe
W-7500 Karlsruhe, FRG
moer@ira.uka.de

26. Christian Rich
Institut für Infomationssysteme
ETH Zürich
ETH-Zentrum

CH-8092 Zürich, Switzerland
rich@inf.ethz.ch

27. Arnie Rosenthal
Xerox Advanced Information Technology
Four Cambridge Center
Cambridge, MA 02142, USA
arnie@xait.xerox.COM

28. Hans-Jörg Schek
Institut für Informationssysteme
ETH Zürich
CH-8092 Zürich, Switzerland
schek@inf.ethz.ch

29. Harald Schöning
Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
W-6750 Kaiserslautern, FRG
schoenin@informatik.uni-kl.de

30. Marc Scholl
Institut für Informationssysteme
ETH Zürich
CH-8092 Zürich, Switzerland
scholl@inf.ethz.ch

31. Bernhard Thalheim
Institut für Informatik
Universität Rostock
Albert-Einstein-Str. 21
O-2500 Rostock, FRG
thalheim@informatik.uni-rostock.dbp.de

32. Gottfried Vossen
Lehrstuhl fur Angewandte Mathematik
RWTH Aachen
Ahornstr. 55
W-5100 Aachen, FRG
vossen@informatik.rwth-aachen.de

Bisher erschienene und geplante Titel:

W. Gentzsch, W.J. Paul (editors):
Architecture and Performance, Dagstuhl-Seminar-Report; 1, 18.-20.6.1990; (9025)

K. Harbusch, W. Wahlster (editors):
Tree Adjoining Grammars, 1st. International Worshop on TAGs: Formal Theory and Application, Dagstuhl-Seminar-Report; 2, 15.-17.8.1990 (9033)

Ch. Hankin, R. Wilhelm (editors):
Functional Languages: Optimization for Parallelism, Dagstuhl-Seminar-Report; 3, 3.-7.9.1990 (9036)

H. Alt, E. Welzl (editors):
Algorithmic Geometry, Dagstuhl-Seminar-Report; 4, 8.-12.10.1990 (9041)

J. Berstel , J.E. Pin, W. Thomas (editors):
Automata Theory and Applications in Logic and Complexity, Dagstuhl-Seminar-Report; 5, 14.-18.1.1991 (9103)

B. Becker, Ch. Meinel (editors):
Entwerfen, Prüfen, Testen, Dagstuhl-Seminar-Report; 6, 18.-22.2.1991 (9108)

J. P. Finance, S. Jähnichen, J. Loeckx, M. Wirsing (editors):
Logical Theory for Program Construction, Dagstuhl-Seminar-Report; 7, 25.2.-1.3.1991 (9109)

E. W. Mayr, F. Meyer auf der Heide (editors):
Parallel and Distributed Algorithms, Dagstuhl-Seminar-Report; 8, 4.-8.3.1991 (9110)

M. Broy, P. Deussen, E.-R. Olderog, W.P. de Roever (editors):
Concurrent Systems: Semantics, Specification, and Synthesis, Dagstuhl-Seminar-Report; 9, 11.-15.3.1991 (9111)

K. Apt, K. Indermark, M. Rodriguez-Artalejo (editors):
Integration of Functional and Logic Programming, Dagstuhl-Seminar-Report; 10, 18.-22.3.1991 (9112)

E. Novak, J. Traub, H. Wozniakowski (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 11, 15-19.4.1991 (9116)

B. Nebel, C. Peltason, K. v. Luck (editors):
Terminological Logics, Dagstuhl-Seminar-Report; 12, 6.5.-18.5.1991 (9119)

R. Giegerich, S. Graham (editors):
Code Generation - Concepts, Tools, Techniques, Dagstuhl-Seminar-Report; 13, 20.-24.5.1991 (9121)

M. Karpinski, M. Luby, U. Vazirani (editors):
Randomized Algorithms, Dagstuhl-Seminar-Report; 14, 10.-14.6.1991 (9124)

J. Ch. Freytag, D. Maier, G. Vossen (editors):
Query Processing in Object-Oriented, Complex Object, and Nested Relation Databases, Dagstuhl-Seminar-Report; 15, 17.-21.6.1991 (9125)

M. Droste, Y. Gurevich (editors):
Semantics of Programming Languages and Model Theory, Dagstuhl-Seminar-Report; 16, 24.-28.6.1991 (9126)