Dagstuhl Workshop
Performance Enhancement in Object Bases
(Leistungssteigerung in Objektbanken)
April 1–4, 1996

U. Dayal[1] *A. Kemper*[2] *G. Moerkotte*[3] *G. Weikum*[4]

[1]Hewlett-Packard Labs.
1501 Page Mill Road, 3U-4
Palo Alto, CA 94304, USA
dayal@hpl.hp.com
Telephone: (415)-857-8431

[2]Universität Passau
Lehrstuhl für Informatik
94030 Passau, Germany
kemper@*db.fmi.uni-passau.de*
Telephone: 0851/509-3060

[3]Universität Mannheim
Lehrstuhl für Praktische Informatik III
68162 Mannheim, Germany
moer@pi3.informatik.uni-mannheim.de
Telephone: 0621/292-5403

[4]Universität des Saarlandes
Lehrstuhl für Informatik
66041 Saarbrücken, Germany
weikum@cs.uni-sb.de
Telephone: 0681/302-4786

Object-oriented database systems (or object base management systems, OBMS) are envisioned as the next-generation information technology. The object-oriented model provides more advanced functionality than the relational model: the object model integrates the *structural* and the *behavioral representation* in one coherent schema (classes or types) whereas the relational model is limited to the structural representation of data. However, despite the functional advantage, in order to become widely accepted as the encompassing information repository the object base management systems have to exhibit a performance that is comparable—or even superior—to the currently predominant relational database systems. No user is willing to trade performance for functionality.

Therefore, in this seminar a wide range of issues related to enhancing the performance of object-oriented database systems were addressed. The seminar brought together leading researchers from different areas of optimization techniques in the object-oriented database development. Various issues from the following research areas were addressed during the seminar.

Query Optimization The ability to process declarative queries has been one of the prevailing strengths of relational DBMSs, since their early days. Since OBMSs support the functionally more powerful object models, they need query processing engines with far greater abilities. Subtopics relevant to query optimization in OBMSs are query optimizer architectures, solutions to special query classes, exploration of object-oriented concepts, and cost models.

Method Optimization A difficult problem in building an OBMS is to deal with methods, i.e., the behavioral component of data. Despite its relevance, there has been only slow progress in this research area. So far, methods have been considered as black boxes by query optimizers. Even worse, it is usually difficult even to attach precise costs to methods. As a result, they have been typically treated as unoptimizable and unpredictable components of a query.

Run Time System Optimization Typical design decisions concern buffer management, object representation, organization of client/server communication, concurrency control, recovery, prefetching objects and/or pages, etc. The evaluation of existing strategies for run time optimization, and the development of new strategies, is the thrust of research within this topic.

Fast Access Methods Since object models and query languages are more complex than their relational counterparts, there are some new issues in the design of access methods for OBMSs. Typical examples of issues to be discussed under this topic are (path) indexes, clustering, and declustering, and their effectiveness in enhancing OBMS performance.

Physical Object Base Design All the above techniques for enhancing performance can be annihilated by poor physical object base design. For instance, if the user does not define appropriate indexes or clustering, or uses too many indexes on the wrong attributes or paths, then the performance of the OBMS degrades drastically. Despite the obvious relevance of good physical design, there has been very little research on methodologies and tools for physical object base design.

Distributed Object Management Due to the increased interconnectivity of computers, individual objects or collections of objects may be physically distributed. In such a scenario, applications will typically require the coordinated execution of queries or transactions over the distributed object base. Several optimization techniques relevant to distributed OBMSs were discussed.

Internet/WWW-Applications Database systems—and in particular, object-oriented databases—are more and more used as the information repositories of the Internet. This results in new requirements with respect to query facilities and performance tuning.

The workshop featured 17 presentations covering the above topics and four working groups. The abstracts of the presentations and summaries of the four working groups comprise this report.

The organizers would like to thank all participants for making this workshop a success—and a pleasant experience.

Umeshwar Dayal Alfons Kemper Guido Moerkotte Gerhard Weikum

Schedule of the Seminar

	Monday	Tuesday	Wednesday	Thursday
morning 9:00–12:15	talks	talks	working group	working group
afternoon 2:00–6:15	talks	talks	outing	summary

Friday being a holiday, this was only a four-day seminar.

Scheduled Presentations

1. Monday Morning (Physical Design and Query Optimization)
   chair: Guido Moerkotte
   (a) B. Seeger: Low-Level Optimization in Database Systems: A Case for Multi-Page Requests
   (b) T. Risch: Efficient Processing of Object-Oriented Queries with Late Bound Functions
   (c) S. Chaudhuri: Optimizing Queries with Methods
   (d) K. Shim: Optimizing Queries with Group-Bys.
   (e) S. Cluet The X-Inst Family of Algebraic Operators
2. Monday Afternoon (Query Optimization)
   chair: Umesh Dayal
   (a) M. Scholl: Optimization in CROQUE
   (b) T. Grust: Monoid Comprehensions as a Target for the Translation of OQL
   (c) T. Özsu: An extensible query optimizer architecture for object base management systems
   (d) Y. Ioannidis: Improved Histograms for Selectivity Estimation of Range Queries
3. Tuesday Morning (Distributed and Parallel Object Bases)
   chair: Gerhard Weikum
   (a) B. Mitschang: The KRISYS Approach to Advanced Data Processing: Parallelism and Client-based Processing
   (b) A. Dogac: Dynamic query optimization in distributed objects platforms
   (c) D. Kossman: Cache Investments Strategies for Distributed Object Bases
   (d) M. Rys/H.-J. Schek: Mapping an object model (COCOON) onto a relational multi-processor database system
4. Tuesday Afternoon (Architectural and Language Issues)
   chair: Alfons Kemper

(a) R. H. Güting: The SECONDO Architecture – Constructing Modular Database Systems

(b) P. Boncz/M. Kersten: Runtime System Organisation in Monet

(c) P. Apers/A. Wilschut: Implementation of an object algebra on Monet.

(d) S. Blott: An Abstract-Object Storage System

Working Groups

- Query Optimizer Benchmarks / Cost Model Verification
  - Moderator: Yannis E. Ioannidis
- Caching and Client-Server Databases
  - Moderator: Donald Kossmann
- Database, Text, Multimedia and Web
  - Moderator: Surajit Chaudhuri
- Query Optimizers for Object Database Systems
  - Moderator: Tamer Özsu

Abstracts of the Talks

Low-Level Optimization in Database Systems: A Case for Multi-Page Requests

Bernhard Seeger

During the last decade several approaches have been emerged to avert a threatening I/O crisis. One of these approaches is termed multi-page requests also known as set-oriented I/O and bulk I/O. A multi-page request is an I/O request that retrieves several pages from disk where the order does not count. As a rule, the pages are located close to each on disk, e.g. on a cylinder, but contiguity is not required. This property of a multi-page request actually distinguishes our definition from most of the previous ones (which explicitly assume contiguity). A multi-page request is performed without interruption of other requests. Thus, the processing time of a multi-page request mainly depends on the read/write schedule that determines the order in which the target pages are retrieved from disk.

In this talk, we first present a simple algorithm for computing a nearly optimal schedule of a multi-page request. Then, the expected cost of the schedules is analyzed by using a pure analytical model. In addition to the disk geometry, the analysis takes into account the four major cost components (seek time, rotational delay, transfer time and head switch time) which occur when pages are retrieved from a magnetic disk. The derived cost function only depends on two parameters: the number of required pages and the degree

of clustering of the underlying file. The results from an implementation on a real disk confirmed the analytical model and our cost function. Thus, the cost functions can also be used in a query optimizer when operations on a higher level exploit the technique of multi-page requests. Moreover, the cost function demonstrates that significant performance improvements can be achieved by using multi-page requests when the required pages are read according to a well-computed schedule. In addition to the response time of a query exclusively performed in the system, we also examine the impact of multi-page requests on the average response time when multiple queries are concurrently performed at a time. Results of an experimental performance comparison demonstrate that the average response time can considerably be improved on a highly loaded system by orders of magnitude when multi-page requests are used in comparison to reading the required pages one at a time.

Processing Object-Oriented Queries with Invertible Late Bound Functions

Staffan Flodin and Tore Risch

New demands are put on query processing in Object-Oriented (OO) databases to provide efficient and relationally complete query languages. A flexible OO data model requires overloading and late binding of function names. Relational completeness requires capabilities to handle queries where functions are inverted, i.e. where it is possible to select those objects y that satisfies fn(y)=x where x is known. A system that supports both late binding and inverted functions must be able to solve fn(y)=x for a given x and unknown y when fn is late bound, i.e. the resolvent (implementation of a function name) to apply on y is selected based on the type of y. This combination of late binding and inverted function calls require novel query processing capabilities to fully utilize indexes referenced in late bound function calls. We present an approach to the management of late binding in query processing. The main result is a query processing method where late bound function calls are efficiently executed and optimized for both inverted and regular execution. The proposed solution is based on substituting each late bound function call in the execution plan with a special function, DTR, which dynamically selects the actual resolvent to call. We define the inverse of DTR and its correctness. We show a dramatic execution time improvement by making DTR invertible and by defining its cost model for query optimization. The improvements are verified by performance measurements.

Optimizing Queries with Methods

Surajit Chaudhuri

The problem of optimizing queries with methods can be broken down in two parts. (1) Extend the optimization algorithms (2) Develop cost models for the methods. In view of the fact that there has been some past work in (1), the most of this talk focuses on (2). The central message of the talk is to stress that there is unlikely to be a robust cost model for arbitrary methods. Rather, what is needed are domain-specific cost models. The talk illustrates how the traditional cost model works well for a class of methods but falls apart in more complex domains. A specially important requirement is to capture

interrelationships of methods in a quantitative way. We conclude by arguing that a scalable architecture for optimizing queries with methods will be 2-tiered, with domain-specific optimizers at the bottom tier and a generic relational-like optimizer as the top tier.

Evaluating Queries with Generalized Path Expressions

Vassilis Christophides, Sophie Cluet, Guido Moerkotte, Jérôme Siméon

In the past few years there has been a growing interest towards query languages featuring generalized path expressions(GPE). With these languages, one may issue queries without exact knowledge of the schema. A generalized path expression queries schema and instance at the same time. Although very useful for standard database applications, these languages are essential to new applications dedicated, for instance, to structured documents.

There have been some proposals for evaluating generalized path expressions. However, these proposals are rather naive. We present an algebraic way of dealing with GPE's in the object oriented context. We introduce an extended object algebra and show the many advantages of this clean and flexible approach.

So far, the technique for evaluating generalized path expressions is to find type information, rewrite the query accordingly, using unions or disjuncts, and, from then on, perform standard optimization. The main drawbacks of this technique are an exponential input for the optimizer (as many disjuncts/unions as possible paths), no flexibility (fixed treatment), many redundancies and intermediate results.

Our proposal consists in extending an object algebra with operators dealing with path at the schema and instance levels. The result is more optimization opportunities for less complexity and less intermediary results. Further, this approach offers many new optimization possibilites and notably allows a flexible use of full text indexes to evaluate queries (either standard of featuring GPE's).

The KRISYS Approach to Advanced Data Processing

Bernhard Mitschang

Co-Authors:

- S. Dessloch IBM Germany, Boeblingen, Germany
- T. Haerder University of Kaiserslautern, Kaiserslautern, Germany
- N. Mattos IBM Data Base Technology Institute, San Jose, CA, US
- J. Thomas University of Kaiserslautern, Kaiserslautern, Germany

Advanced data models together with their powerful query and manipulation languages have already proven to be essential for systems that support advanced applications such as engineering and knowledge-based application systems.

Over the last couple of years, we have been developing such an advanced database systems - the KRISYS prototype - that shows a number of advanced data processing

capabilities that have already proven their usability within experimental application scenarios. Of course, there has been a number of iterations to the KRISYS prototype mostly triggered by the results achieved by means of these application experiments.

Now, since we know that a stable point in the KRISYS system development has been reached, it is worthwhile to report on both its approved data processing capabilities as well as on the experiences gained and the lessons learned. The net effect of this technical discussion will bring up a number of important aspects w.r.t. advanced data processing that are of significant general importance as well as of general applicability to data processing systems targeted for client/server architectures.

In particular the following KRISYS concepts have been discussed in more detail: - Query Processing at client and at server site - Client infrastructure for main-memory query processing - Run-time query optimization - Extensibility at different levels of query processing.

Optimizing Queries with Group-Bys

K. Shim

In existing relational database systems, processing of group-by and computation of aggregate functions are always postponed until all joins are performed. In this talk, we present transformations that make it possible to push group-by operation past one or more joins and can potentially reduce the cost of processing a query significantly. We explain how the traditional System-R style optimizers can be modified by incorporating the greedy conservative heuristic that we developed for single block SQL. We then show how complex queries, with aggregates, views and nested subqueries can be optimized efficiently in a cost-based fashion. The crux of our solution is a careful treatment of group-by and aggregation operators that occur among views.

Optimization in CROQUE

Marc Scholl

In the framework of the CROQUE project [CROQUE stands for *C*ost- and *R*ulebased *O*ptimization of object-oriented *QUE*ries, a joint project of the University of Konstanz and the University of Rostock, sponsored by the German Research Association (DFG).], we focus our attention on the development of optimization techniques for object databases. Central topics of our work are rule-based rewritings of object-oriented queries, cost-based selections of evaluation mechanisms for those queries, and optimization of the physical design of object databases with respect to a given application schema.

The goal of the project is the development of an optimizing, object-oriented query evaluation system with three characteristics:

- a descriptive, object-oriented query language:
  Descriptive ODB queries, expressed in CROQUE-OQL/OML – our variant of ODMG-OQL/OML – over the logical ODB schema are transformed and optimized towards efficient internal execution plans by the CROQUE query optimizer.

- logical data independence:
  Provide a clean separation of conceptual and physical design, this is not yet realized in most object-oriented database systems (such as, e.g., ObjectStore, Versant, Ontos, and POET). The logical database schema is defined in CROQUE-ODL, our variant of the ODMG 1.2 object definition language. During physical database design, the database administrator can use an estimated or observed transaction load (i.e., a set of transaction programs with their frequencies) to optimize the layout of internal storage structures. Particular emphasis lies on exploiting physical designs that can differ significantly from the logical schema's view of data to due: partitioning, clustering, (partial) replication of DB objects, ...

- a cost- *and* rule-based optimizer
  translating descriptive queries and the update operations from the conceptual level into efficient execution plans depending on the chosen storage structures.

More information on the CROQUE project as well as references to on-line documents are available from `http://www.informatik.uni-konstanz.de/~gluche/croque/croque.html`

Monoid Comprehensions as a Target for the Translation of OQL

Torsten Grust and Marc H. Scholl

The rich type models that make object-oriented data models superior to their relational antecedents have an impact on the query formalism that is needed to capture today's modern object query languages, like ODMG's OQL. Instead of reintroducing the classical object algebra operators for each of the bulk type constructors like *set*, *bag*, and *list* — which led to rather intractable algebras in recent approaches, especially suffering from type conversion operators — the concept of a *monoid* [2] can take the role of what the *set* played in the relational model. These simple algebraic structures, obeying a few simple laws, allow for a uniform treatment of collections and scalars. They therefore account for the intermix of bulk operations and general purpose computations (e.g. arithmetics) that modern query languages feature. *Comprehensions* provide a convenient way to describe operations carried out on monoids, thus providing us with a powerful object calculus. ODMG OQL can be mapped to this calculus in a straightforward manner [1].

The problem of deriving efficient execution plans for these calculus expressions has been tackled by recent work that promotes *normalization by unnesting* [2]. However, these approaches do not overcome the nested-loop nature the comprehensions imply. In contrast, we propose to translate OQL into a *hybrid mix* of calculus and algebra expressions, letting each formalism handle the parts of the queries it can cope best with. We derive query plans that employ the strenghts of objects algebras when it comes to set-based computations, joins, grouping, etc., and at the same time can reason about quantification, arithmetics, and type conversion expressed by calculus expressions [1]. In pure algebraic approaches, the latter were not captured by the formalism and as a consequence have been "black boxes" that were simply passed around during the rewriting phase of rule-based optimizers. The algebra and calculus we use are tailored to support the interplay between both, e.g. in terms of exchanging selection predicates.

We claim, that the proposed framework captures OQL-like query language completely and allows the derivation of more efficient execution plans, employing sophisticated join

techniques, like *nestjoins* and *bypass joins*.

References

[1] Torsten Grust and Marc H. Scholl. Translating OQL into Monoid Comprehensions — Stuck with Nested Loops? Technical Report 3/1996, Faculty of Mathematics and Computer Science, Database Research Group, University of Konstanz, March 1996.

[2] Leonidas Fegaras and David Maier. Towards an Effective Calculus for Object Query Languages. In *ACM SIGMOD Int'l Conference on Management of Data*, May 1995.

An Extensible Query Optimizer Architecture for Object Base Management Systems

Tamer Özsu

The extensible query optimization has been studied within the context of relational database management systems (DBMSs) where new optimization rules and, sometimes, cost functions can be defined to guide the optimization process. Extensibility becomes even more important in the case of object database management systems (ODBMSs). The important reasons are the following: (1) The optimizer should be easily customizable since the application domains to be supported by ODBMSs have differing query processing and optimization requirements; (2) Since there is no consensus about an object algebra (even though there are many proposals) and since ODBMSs allow application-specific algebra operators to be defined, the optimizer should be customizable and should treat user defined operators uniformly; (3) The optimization techniques for ODBMSs are not fully developed and the alternatives are not completely understood with improvements occurring regularly, requiring the optimizer to easily incorporate these changes. These requirements point to a need to be able to build systems which can accommodate extensions along all three dimensions of the query optimization process: transformation rules (which define the search space), search algorithms, and cost functions.

We describe an extensible query optimizer for the TIGUKAT ODBMS. TIGUKAT is a uniform behavioral system that models every system component as a first-class object. Consistent with this philosophy, we model every component of the optimizer as a first-class object, providing ultimate extensibility. We describe the optimizer architecture and how the optimizer components are modeled as extensions of a uniform type system.

All publications on TIGUKAT can be obtained from our Web page at

`http://www.cs.ualberta.ca/~database`

Histogram-based Estimation in Databases

Yannis E. Ioannidis[1]

[1] Various parts of the work summarized here have been done in collaboration with Stavros Christodoulakis, Peter Haas, Vishy Poosala, and Eugene Shekita.

Work partially supported by the National Science Foundation under Grant IRI-9113736 and Grant IRI-9157368 (PYI Award) and by grants from DEC, IBM, HP, AT&T, Oracle, and Informix.

Several modules of a database system require estimates of (sub)query result sizes, e.g., query optimizers or sophisticated user interfaces providing such estimates as feedback to users before a query is actually executed. These estimates usually have errors, which could become large and significantly reduce the effectiveness of these modules. Thus, close-to-accurate size estimation is of critical importance.

In practice, most database systems (e.g., DB2, Informix, Ingres, Microsoft SQL Server, Sybase) base their query-result size estimation on some type of *histograms*. Histograms approximate the frequency distribution of an attribute by grouping attribute values into "buckets" (subsets) and approximating true attribute values and their frequencies in the data based on summary statistics maintained in each bucket.

In our work, we have identified several key properties that characterize histograms and determine their effectiveness in query result size estimation [5]. These properties are mutually orthogonal and form the basis for a general taxonomy that captures all existing histogram types. Moreover, we have introduced novel techniques for several of the taxonomy dimensions, and have then derived new histogram types by combining these techniques in effective ways. We have studied these histograms for several estimation problems (query result size of range selections [5] and equality selections and joins [1, 2, 3], but also query result distribution of equality joins [4]). Through a series of analytical results and experimental evaluations (using both synthetic and real-life data), we have identified a small number of the newly proposed histogram types as the most effective for these estimation problems, much better than the traditional ones currently used in practice. We have also provided efficient histogram construction methods that first sample the database to construct an approximate distribution of the data values and then apply the appropriate algorithm to construct the histogram. Finally, we have refined the kinds of information that is maintained within each histogram bucket in ways that have improved the histogram's accuracy (compared to the traditional kinds of information).

We are currently studying the effectiveness of various types of histograms in capturing attribute value correlation, and in approximating the result size of more complex queries, e.g., those involving aggregates. Our hope is that a small number of easily implementable histogram types will emerge as the clear winners for all estimation problems, so that systems will only need to deal with (one of) these.

References

[1] Y. Ioannidis. Universality of serial histograms. In *Proc. 19th Int. VLDB Conference*, pages 256–267, Dublin, Ireland, August 1993.

[2] Y. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM TODS*, 18(4):709–748, December 1993.

[3] Y. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proc. of the ACM-SIGMOD Conf. on the Management of Data*, pages 233–244, San Jose, CA, May 1995.

[4] V. Poosala and Y. Ioannidis. Estimation of query-result distribution and its application in parallel-join load balancing. In *Proc. 22nd Int. VLDB Conference*, Bombay, India, September 1996 (to appear).

[5] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proc. of the ACM-SIGMOD Conf. on the Management of Data*, pages 294–305, Montreal, Canada, June 1996.

Dynamic Query Optimization on DOM Platforms

Asuman Dogac

A Distributed Object Management (DOM) architecture, when used as the infrastructure of a multidatabase system, not only enables easy and flexible interoperation of DBMSs, but also facilitates interoperation of the multidatabase system with other repositories that do not have DBMS capabilities. This is an important advantage, since most of data still resides on repositories that do not have DBMS capabilities.

In this paper, we describe a dynamic query optimization technique for a multidatabase system, namely MIND, implemented on a DOM environment. Dynamic query optimization, which schedules intersite operations at run-time, fits better to such an environment since it benefits from location transparency provided by the DOM framework. In this way, the dynamic changes in the configuration of system resources such as a relocated DBMS or a new mirror to an existing DBMS, do not affect the optimized execution in the system. Furthermore, the uncertainty in estimating the appearance times (i.e., the execution time of the global sub-query at a local DBMS) of partial results are avoided because there is no need for the dynamic optimizer to know the logical cost parameters of the underlying local DBMS.

In scheduling the intersite operations a statistical decision mechanism is used. We also consider the schema integration information to make room for further query optimization. For this purpose, an algorithm is presented that forms global query graphs by taking the schema integration information into account which is then used by the dynamic query optimizer. The proposed scheme tries to exploit the inherent parallelism in the system as much as possible.

The performance of the developed method is compared with two other most related techniques and the results of the experiments indicate that the dynamic query optimization technique presented in this paper has a better performance.

Cache Investment Strategies for Distributed Object Bases

Donald Kossmann and Michael J. Franklin

In traditional distributed database systems, caching decisions and query optimization are carried out independently. This work shows that there is a circular dependency between caching and query optimization and that, therefore, traditional query optimizers should be extended to take caching into account. Depending on query optimization, the state of the cache of a site of a distributed database system can change significantly: if the optimizer places operators of a query at a site, the data used in those operators can be cached at that site subsequently; on the other hand, the contents of the cache of a site remains unchanged if no operators are executed at the site. Because of this dependency between caching and query optimization, it is often beneficial to generate a sub-optimal plan for a query. Such a sub-optimal plan initiates the caching of data at a site and,

thus, potentially reduces the cost of future queries if the data is used in future queries. If the data is not used in future queries, however, the caching of the data should not be initiated and the optimizer should not generate a sub-optimal plan. To control when the optimizer should generate a plan that initiates the caching of data at a site, a *cache investment policy* is required. We present four alternative cache investment policies and compare their performance in an extensive simulation study.

Implementing an Object Model (COCOON) on Top of a Relational Multi-Processor Database System

Michael Rys and Hans–Jörg Schek

We map an object model to a commercial relational multi-processor database system using extensive replication and view materialisation to provide fast retrieval. To speed up the complex update operations, we employ intra-transaction parallelism by breaking such an operation down into shorter relational operations. These operations are executed as parallel subtransactions of the object level's update operation. In other words, we exploit inter-transaction parallelism on the relational level to achieve intra-transaction parallelism on the object level.

To ensure the correctness and recoverability of the operation's execution, we use multi-level transaction management as the parallelisation framework. In addition, we minimise the resulting overhead for the logging of the compensating inverse operation required by the multi-level concept by logging the compensation for nonderived data only.

We show that we can efficiently maintain the replicated data and materialised views in the context of large classification structures by presenting a performance evaluation of an exemplary set-oriented update statement.

The SECONDO Architecture – Constructing Modular Database Systems

Ralf Hartmut Güting

The talk describes the SECONDO architecture for database systems. The design pursues the following goals:

- Develop a database system (frame) without a data model.
- Describe the data model to the system in a high-level specification language.
- Describe a query processing subsystem to the (frame) system in the same language.
- Describe also optimization in this language.
- Design the query processing subsystem as a collection of algebras, and implement each algebra as an *algebra module*.
- Design clean, simple, easy-to-use interfaces and tools for the implementation of algebra modules, so that research groups anywhere in the world can independently encapsulate and make available their latest research results (prototypes) in the form

of algebra modules, and so contribute to an open library of DBMS query processing modules.

As a high-level specification language, second-order signature is used. We discuss the overall architecture, the set of commands executed by the SECONDO system, the structure of algebra modules, some possible modules one might design, and interfaces between modules.

Building an OO system on top of Monet

Peter A. Boncz, Martin Kersten

Summary

We discuss how Monet, a novel multimodel database system, can be used to efficiently support OODB applications. We show how Monet's offbeat view on key issues in database architecture provided both challenges and opportunities in building a high-performance ODMG-93 compliant Runtime System on top of it.

We describe how an OO data-model can be mapped onto Monet's decomposed storage scheme while maintaining physical data independence, and how OO queries are translated into an algebraic language. A generic model for specifying OO class-attribute traversals is presented, that permits the OODB to algebraicly optimize and parallelize their execution.

To demonstrate the success of our approach, we give OO7 benchmark results of our Runtime System for both the standard pointer-based object navigation, and our declarative model based on a path-operator traversal library.

Introduction

Engineering design and CASE are the prototypical database applications that require the database system to support complex and evolving data structures. Queries often involve -hierarchical- traversals and have to be executed with high performance to satisfy the requirements posed by an interactive application.

OODBs have been identified as the prime vehicle to fulfill these tough demands. It is in these application domains that traditional RDBMSs suffer most from the impedance mismatch, and fail to deliver flexibility and performance. In recent years several – commercial – OODBs have entered the marketplace. Since "performance" in CAD/CAM or CASE applications has many faces, the OO7 benchmark was introduced as a yardstick for their success. It measures traversal-, update- and query evaluation performance for databases of differing sizes and object connectivity. The results published [1] indicate room for further improvement and a need for more effective implementation techniques.

This article describes how we tackled the OO7 functionality with our ODMG-93 compliant Runtime System called MO_2 [4] that was developed on top of Monet [3]. Monet is a novel database kernel that uses the Decomposed Storage Model (DSM [5]) because of its effectiveness in main-memory dominant environments. Through its use of virtual-memory techniques and operating system facilities for buffer management, Monet has been proven capable of handling both small and huge data-volumes efficiently [2].

Monet is a *multimodel* system; this means that its data can be viewed simultaneously in a relational, binary set-algebraic, or object-oriented manner. The MO_2 system is put

at the task of translating between Monet's DSM- and the object-oriented data-model. This translation provides many opportunities for optimization, such as the *lazy attribute fetching* technique employed in MO_2.

From the viewpoint of an OODBS, traversals specified in a persistent programming language like C++, result in a waterfall of individual object-fetches optimization cannot take place anymore. Helped by the physical data independence present in the MO_2 system, we managed to improve on this by offering a generic model for specifying complex traversals at a high level of abstraction. Traversals specified with this model can seamlessly be integrated with set-oriented query optimization and parallelization, for efficient execution on Monet.

References

[1] M. Carey and D. J. DeWitt and J. F. Naughton. The DEC OO7 Benchmark. In *Proc. ACM SIGMOD Conference*, Washington, DC, USA, May 1993.

[2] P. A. Boncz and W. C. Quak and M. L. Kersten. Monet and its Geographical Extensions: A Novel Approach to High Performance GIS Processing. In *Proc. EDBT'96 Conference*, Avignon, France, March, 1996.

[3] P. A. Boncz and M. L. Kersten. Monet: An Impressionist Sketch of an Advanced Database System. In *Proc. IEEE BIWIT workshop*, San Sebastian, Spain, July, 1995.

[4] C. A. van den Berg und A. van den Hoeven. Monet meets OO7. In *OO Database Systems Symposium of the Engineering Systems Design and Analysis Conference*, Montpellier, France, July, 1996.

[5] G. Copeland and S. Khoshafian. A Decomposed Storage Model. In *Proc. ACM SIGMOD Conference*, Austin, TX, May, 1985.

An Open Abstract-Object Storage System

Stephen Blott and Lukas Relly and Hans-Jörg Schek

Database systems must become more open to retain their relevance as a technology of choice and necessity. Openness implies not only databases exporting their data, but also exporting their services. This is as true in classical application areas as in non-classical (GIS, multimedia, design, etc).

This paper addresses the problem of exporting storage-management services of indexing, replication and basic query processing. We describe an abstract-object storage model which provides the basic mechanism, '*likeness*', through which these services are applied uniformly to internally-stored, internally-defined data, and to externally-stored, externally-defined data. Managing external data requires the coupling of external operations to the database system. We discuss the interfaces and protocols required of these to achieve correct resource management and admit efficient realisation. Throughout, we demonstrate our solutions in the area of semi-structured file management; in our case, geospatial metadata files.

Summaries of the Working Groups

Query Optimizer Benchmarks / Cost Model Verification

Moderator: Yannis E. Ioannidis

This short report summarizes the discussions and recommendations of the working group on "Query Optimizer Benchmarks and Cost Model Verification". It does not propose an actual benchmark, but provides some guidelines and requirements for what such a benchmark should look like. Designing a benchmark that follows these guidelines is left for the (not so distant?) future.

Query Optimizer Benchmarks

The goal of all existing database benchmarks, e.g., TPC-{C,D}, Wisconsin, Sequoia, or 007, is to measure the overall performance of complete database systems. Since it determines what kind of a plan is executed for any given query, the query optimizer module plays a major role in how systems perform, so in some sense its effectiveness is also tested (at least partially) by traditional benchmarks. Nevertheless, given the importance of query optimizers, a standard benchmark that would measure the effectiveness and efficiency of stand-alone query optimizers is very desirable. The following subsections elaborate on the architectural requirements, the design requirements, and the measures of success that are proposed for such a benchmark.

Architectural Requirements

An abstract architecture of a query optimizer is shown in Figure 1. The planner is the

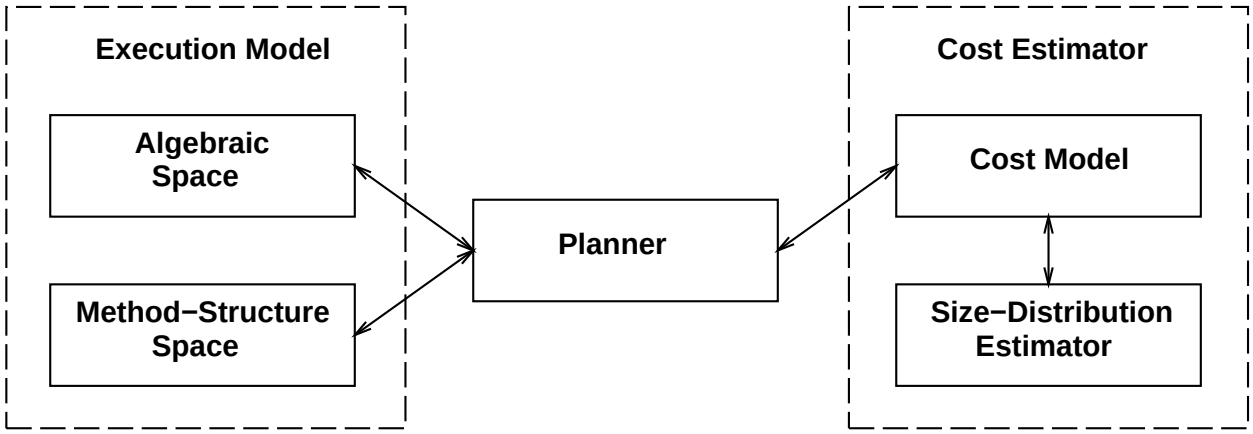

Figure 1: Abstract architecture of a query optimizer

heart of the system that searches all the alternative access plans for a given query. The algebraic and method-structure spaces define the execution model, which captures the primitive operators of the underlying execution engine (e.g., the join is in the algebraic space while nested-loops or B+-tree accesses are in the method-structure space). The cost model is a set of formulas that approximate the cost of each primitive operator in the execution model. The size/distribution estimator provides estimates for the sizes of

(sub)query results and the distributions of values in them, which are the most important parameters of the cost model formulas. These two modules form the cost estimator, which provides the actual estimates for the cost of query (sub)plans.

Given the above architecture of an optimizer, the architecture of the benchmark itself should satisfy the following requirements:

- *Genericity/Modularity/Extensibility*: The benchmark should be generic and applicable to a wide variety of optimizers. It should be modular, so that pieces of it can be easily replaced by others if needed. It should also be extensible so that if some optimizer supports features not included in the basic benchmark, they can be added smoothly by that optimizer's developers for testing.

- *Execution model*: The execution model of an optimizer is determined by modules of the database system that are outside of the optimizer itself, essentially the execution engine. To facilitate comparisons of optimizers independent of the corresponding underlying execution engines, the benchmark should provide an abstract execution model that includes all the commonly found operators, e.g., scanning a file, probing an unclustered B+-tree index, nested-loops between two files, etc. Of course, given the modularity of the benchmark, one should be able to plague in a different execution model instead of the standard one if so desired. Even in that case, however, it may be desirable to translate the final access plans into the standard model so that their cost can be reported in a standard form.

- *Cost estimator*: Although the cost estimator is an integral part of the optimizer, one often wants to compare optimizers without considering any differences in the cost estimates that are generated for the various access plans. This may be the case for just the cost model, just the size/distribution estimator, or both. To facilitate such comparisons, the benchmark should provide a standard cost model for its execution model and also a standard size/distribution estimator. Again, given the modularity of the benchmark, it should be easy to use these or any other alternatives that optimizers may provide.

- *Cost units*: The actual execution cost of a plan depends heavily on the speed of the various resources of the system it runs on, e.g., CPUs and disks. To standardize comparisons across machines, possibly based on published results rather than actual experimentation with all optimizers of interest, the benchmark should offer a standard set of cost units that should be abstract and independent of any system characteristics. Moreover, the costs generated by the formulas in the cost model should be expressed in terms of these units.

Design Requirements

Like in the traditional database benchmarks, the optimizer benchmark should specify the queries that should be optimized and the underlying database (schema and actual data) on which these queries should (hypothetically) be executed. The requirements on these two are discussed below:

- *Queries*: There should be a set of fixed queries that try to capture a wide variety of characteristics. These queries should be realistic and taken from actual applications,

if possible. Using (modified versions of) queries in some general benchmarks, e.g., TPC-D, is also a possibility.

In addition to the fixed set of queries, the benchmark should provide a framework for testing optimizers on random queries. A space of orthogonal query characteristics should be specified, e.g., number of joins, query graph shape, and other such properties, together with specific choices/values in each dimension. During an actual benchmark run, for each combination of choices, a large number of queries would be generated randomly, and the typical behavior of the optimizer on them would be extracted in terms of various statistics, e.g., average and standard deviation.

- *Database*: The schema on which the above queries will be posed should be fixed both at the logical and the physical level. It should be quite large and should include many different features to facilitate the specification of a variety of queries that would test various aspects of the optimizer.

  The data that will presumably populate this schema should also be fixed. Ideally, this data would be stored in some central place from which it could be down-loaded. If this is not possible, however, since no actual database access is necessary for evaluating optimizers, a detailed description of the data would be enough without an actual materialization. Independent of the above, the data should be varied enough to test various aspects of the optimizer, e.g., different sizes of relations, different data distributions in attributes (from uniform to highly skewed), different attribute value correlations, etc.

Measures of success

There are essentially two measures of success for an optimizer operating on a given query:

- *Cost of output plan*: The effectiveness of an optimizer is measured by the cost of the output plan. This should be given in terms of the standard cost units that were mentioned above among the architectural requirements of the benchmark. Although not a requirement, it would be good if some central place stored the best costs that have ever been reported for each benchmark query, so that some sense of the "absolute" effectiveness of an optimizer is obtained. This is really important for optimizing large queries, for which the trully optimal plan cannot be known since they cannot be optimized exhaustively.

- *Cost of optimization*: The efficiency of an optimizer is measured by the time it needs to run. How this should be measured cannot really be standardized, so the actual time (e.g., in seconds) should be reported together with details on the type of machine, size of available memory, and other environment characteristics that may affect the optimizer's performance.

For some optimizers, there is a trade-off between their efficiency and effectiveness (e.g., essentially all those based on randomized search strategies). These optimizers should be run multiple times, each requiring different amounts of time, so that the details of this trade-off can be obtained.

Cost Model Verification

As mentioned above, the cost model of an optimizer includes formulas that approximate the cost of the various operators that are part of its execution model. In principle, on expects that the accuracy or not of this approximation directly affects the effectiveness of the optimizer, as significant errors in the resulting cost estimates may dramatically affect the plan chosen by the optimizer. On the other hand, extreme optimists may claim that even if the errors are large, they usually affect all plans in the same way, so the final output plan is always good. Wherever the truth may lie, it is important to find out how sensitive the optimizer's output is to the accuracy of the cost model. Moreover, assuming that it *is* sensitive in many cases (e.g., it is a single page difference between having a relation fit in main memory and not), it is clearly important to verify the accuracy of a cost model compared to an actual execution engine. Unfortunately, results of such efforts have rarely been reported in the literature, e.g., verification of the System-R and System-R* cost models are rare examples.

For cost model verification, queries need to be optimized and executed, and the estimated costs of the resulting query plans need to be compared with those plans' actual costs. An issue that arises in such an effort is related to whether verification should be done by observing an actual system in operation or through a carefully selected suite of constructed queries/plans on top of synthetic data (as in the benchmark). The advantage of the former approach is that testing concentrates on real-life queries and data. The advantage of the latter approach is that testing can be more focused so that, in the case of large discrepancies between estimated costs and actual costs, the necessary changes to the cost model can be identified easier. Which approach is more effective is unclear.

Another issue is related to the kind of queries/plans that should be used. One possibility is to identify cases that are "hard" to approximate and attempt to derive close-to-accurate cost models for those. The reasoning behind this is that the cost formulas obtained in this effort will hopefully work well for simpler cases as well. Whether this is valid or not is also unclear.

Clearly, a lot more thought needs to be put into cost model verification before some standard methodology can emerge. It's certainly an important topic, however, that should be looked into.

Caching and Client-Server Databases

Moderator: Donald Kossmann

Caching has emerged as a fundamental technique for ensuring high performance in large-scale distributed database and information systems. Caching establishes copies of data at sites; this way, caching reduces communication costs for query processing and helps utilizing the resources of all the sites of a system. Furthermore, caching can adapt to the workload of a site; for example, by using an LRU replacement policy to manage the contents of the site's cache.

Caching has been integrated into existing systems in many different ways. Some systems cache the results of queries (i.e., views) at sites. Other systems cache only unprocessed base data either in the granularity of objects (e.g., tuples or images), pages which can contain several small objects or parts of a large object, or partitions of tables.

Another dimension in which the implementation of systems differ is the cache coherency protocol: possible options are invalidation or propagation-based protocols, or protocols which relax strong cache consistency and serializability in favor of improved performance. Furthermore, data can be cached in a site's main memory or on a site's disk. Another feature which is closely related to the way caching is integrated into a system is whether queries are executed at clients, servers, or both.

The diversity of techniques used to effect caching in existing distributed systems is demonstrated in Table 1. The table characterizes three commercial systems and three research prototypes which were presented at this Dagstuhl workshop. The three commercial systems are Sybase as a representative of the relational camp, O_2 representing the object database camp, and Netscape as the most popular WWW browser. The three research prototypes are Monet (developed at CWI Amsterdam), KRISYS (Kaiserslautern and Munich), and the Abstract Object Storage System (ETH Zürich). All theses systems have fundamentally different architectures. This list of systems could easily be extended finding examples of numerous other client-server architectures.

	Sybase[2]	O_2	Netscape	Monet	KRISYS	AOSS
Unit of Caching	view	page	object	table	view	any
Coherency Protocol	propagation	invalidation	fuzzy	?	propagation	?
Disk Caching	yes	yes	yes	yes	no	yes
Query Execution	server	client	–	both	both	both

Table 1: Characterization of Commercial Systems and Research Prototypes

There are several reasons why existing systems are built in so many different ways; for example, some systems are biased to a particular data model or class of applications. To guide system developers and users in their design choices, several research topics must be addressed:

- The tradeoffs between the alternative approaches to effect caching (e.g., view caching vs. page caching) are not yet fully understood.
- It is not clear which techniques are compatible and can be integrated into the same system; for example, it does not seem to be viable to support view caching using an invalidation-based cache coherency protocol.
- How can flexible systems that adapt to a wide range of applications and system configurations be built?

Database, Text, Multimedia and Web

Moderator: Surajit Chaudhuri

The working group considered the research directions that related database technology to text, multimedia and web management. In order to achieve a better integration with

[2]Sybase differs from the other systems because copies of data are established as part of the physical database design by a system administrator. The other systems support demand-driven caching which is initiated as part of the execution of application programs.

the information retrieval model, it is necessary to support ranking answers in queries. Another question is to study carefully what hooks in the database are necessary for a better integration with a "text-blade" (e.g., using a text blade based on vector-space model). Multimedia query retrieval requires efficient multidimensional index structures in addition to requiring ranked answer set. The web brings new challenges. It presents a schema-less database. It also makes it necessary to reconcile answers from multiple search engines. Perhaps, most exciting is the opportunity to support services beyond querying. An example is the yellow-page service with the need to characterize information provider for content, quality, cost and relevance.

Query Optimizers for Object Database Systems

Moderator: Tamer Özsu

This working group tacked the problem of what needs to go into the design of object query optimizers. Given the short duration that was allocated to the working group meetings, we decided to focus on the requirements for query optimizers. This resulted in the following list:

1. Extensibility.
2. Constrained optimization.
3. Ability to incorporate dynamic optimization.
4. Openness.

One requirement that was discussed at length, but about which there was significant division of opinions was "two level versus one level optimization". Two level optimization refers to the separation of logical optimization from lower level physical optimization. Logical optimization step takes, as input, a "logical" algebra query and produces a "logical" optimization plan which is than operated on by the physical optimizer to produce a "real" execution plan. Proponents of this approach argued that the result is a cleaner system design where some of the optimization can be performed without having to consider the physical organization of data. The latter can be taken into account by a physical optimizer which further optimizes the selected logically optimized query. The opponents pointed to the inherent difficulties of being able to do significant optimization without consideration of the physical data organization. They also pointed to the fact that the "best" logical expression may not lead to the best optimized query once physical data organization is taken into account. The issue was discussed at length, without reaching a final resolution.

Extensibility

Query optimizers can be characterized along three dimensions: search space (i.e., the set of all transformation rules), the search strategy, and the cost model. The participants generally agreed that extensibility of the query optimizer in all its dimensions is required. Many of the existing optimizers and optimizer generators permit the definition of new transformation rules and some even permit the definition of cost functions. However, the

existing systems generally have "built-in" search strategies. Given the differing application environments that object DBMSs are expected to serve, it is essential that optimizers be able to deal with different search strategies. Thus, the need for extensibility in all its dimenstions.

A number of important issues were identified for achieving extensibility. The first has to do with the identification of the building blocks and their organization and representation. Recent literature suggests a number of different architectural frameworks for organizing optimizer components (some of these were discussed at other sessions of the Workshop) and the debate as to which one is superior has not yet been resolved. The second, related, issue has to do with the methodology for putting together these building blocks. The architectural frameworks are not of much help if there isn't an associated methodology that enables developers to specify how to put these pieces together to obtain a working optimizer. The participants identified this as an area which could use more research.

Constrained Optimization

It should be possible to specify and constrain the resources used during query processing. This means that it should be possible to constrain the resources used for optimization and for plan execution. In such an environment, the optimizer should be able to gracefully adjust to the available resources. Some of the current techniques, such as randomized optimization algorithms, enable constraints to be placed on the time taken for optimization. However, this should be extended to the entire query execution process covering all system resources.

Dynamic Optimization

Most optimizers are static, performing their functions during query compilation. In certain circumstances, this may be error prone (propagation of errors) or not possible where run-time information is required. Therefore, the optimizer should be able to incorporate dynamic optimization techniques that allow optimization to be performed at run-time.

A distinction was made between parametric optimization and "real" dynamic optimization. In the former, the static optimizer generates a family of query plans, parameterized with the run time characteristics that are of interest. One of these plans is chosen at run time based on the values of the chosen parameters. In the latter, optimization is performed dynamically at run-time.

Dynamic optimization approaches require both on-line and off-line feedback to be really useful. Feedback includes both the optimization steps that are followed, the execution environment characteristics that are considered and other actions taken by the optimizer.

Finally, it was pointed out that dynamic optimization techniques demonstrate a higher need for the ability to constrain cost of optimization.

Open

The query optimizer architecture should be open, allowing interaction with the user. Openness implies the ability to provide to the optimizer application hints regarding the peculiarities and the specific requirements of the application that issued the query, and

hints on search space, cost model and optimization strategy (e.g., dynamic vs static). These hints could be both positive ("do this") and negative ("don't do this"), perhaps even involving full execution plans. Of course, there needs to be a language to specify all of these hints.

Conclusions

The working group established a number of requirements for query optimizers for ODBMSs. Taken in its entirety, this list of requirements is very ambitious and poses significant challenges, particularly in being able to develop robust and efficient optimizers that fulfill this list. Obviously, there is a lot of work that remains to be done in this area.

On the Humorous Side: Call for research proposals

by: Gerhard Weikum

Bill Gates is thinking about giving you, the distinguished researchers that happen to sit at this dinner table, a modest amount of money for a research and/or development project of your own choice.

Your budget will be 10 million Dollar per year and the timeframe of the project will be 2 years. With this money you can either a) buy 10.000 notebooks, b) 200 GBytes of main memory, c) 2000 kilometers of fibre optical cable, d) 1 million bottles of Chateau Guydeaux red wine, e) pay 100 European graduate students, f) 500 U.S. graduate students, g) 1000 Microsoft engineers, h) 2 consultants from Arthur Andersen, or ... (I hope you got the idea).

You should come up with a sketch of your goals and a plan how to tackle the problems towards these goals. At the end of the 2 years you do not need to have a product or pre-product ready, but you should in any case have a clear vision or plan of how to futher proceed in terms of technology transfer towards a product of application. (You better take into account that Bill Gates is not a particularly patient guy).

You should prepare a five-minute presentation of your proposal, using at most one slide (and no powerpoint presentations or any of the other presentation fads).

You can a) think about this challenge all night, or b) hope for a spontaneous inspiration after three bottles of wine, or c) start working on this tomorrow morning at 9:00 (within your randomly formed group of distinguished researchers). Please have your presentations ready by 10:30 at the latest.

The Winning Proposal: Stinky: a new multi-media I/O device

M.L. Kersten, CWI, The Netherlands[3]
P.M.G. Apers, University of Twente, The Netherlands
J. Gruser, INRIA, France
T. Risch, Linkoping University, Sweden
M. Scholl, University of Konstanz, Germany

Introduction

The dawn of a new era of IT technology is quickly taking shape. Information dissemination for the people has reached the point that text, audio, and video are delivered selectively and on demand.

The research laboratories have already demonstrated the technical feasibility to create *virtual realities* for these media as well. The user (homo sapiens) becomes equipped with devices that mimics unexplored worlds of sensation and opens routes for long-distance courtship.

Yet, an essential ingredient in controlling the human senses is still missing. The ear and eye are dealt with, but the nose has been ignored so far. The role of the nose in our sensorary system plays a key role in detecting smells, which in turn trigger sensations such as mood, taste, and sexual-drive.

The role of smell has become a global market with high return on investment. The perfume industry (a multi billion market) selects odors, superimpose an audio and visual virtual world of macho's, sex-bombs, jungle trotter, and smelly carpenters, so as to open up yet another niche of the market.

The state of the art in chemistry is that you can already order perfume to stimulate the buying drive of humans in shops and most households contain odor producers at various places to disguise the smell of human activity.

It is even safe to predict that in the near future one might order a personalized perfume or that an anti-odors generator infrastructure is installed in France to counter the effects of cigaret smoke without affecting personal freedom.

In the remainder of this document we outline a new Microsoft spinoff company, called *MicroSmell*, which sets out a course to deliver the *MicroSmell* I/O device to enhance the multi-media boxes being sold. The target date to hit the market is Xmas 1998 with a warming up activity during the summer of 1998 by launching the Internet Smell Service.

Project Planning

The generous budget allocated by B. Gates to explore new routes of technology has to be spent according to the table below. A short rationale follows.

[3]This document is also available on the Internet: `http://www.cwi.nl/~mk/stinky/stinky.html`

Activity	Budget
Marketing	4 M
Prototypes	1 M
Software and Research	2 M
Domain experts	2 M
Legal experts	1 M

Given the strong economic power of the perfume industry it is mandatory to start in an early phase to market the concept of personalized smells. Due to lack of past credentials of artificial smells, such a plan should be complemented by an attack on the perfume industry itself. The possible options are to infiltrate this industry with some of our key IT engineers with smell experience of many late hour works, or more aggressively by buying key players in the market. The latter would, however, have a significant effect on our short term revenue balance. A budget of about 2.4 B US$ for a three year attack plan should be reserved.

The MicroSmell prototypes can be configured from off-the-shelf chemical devices. The key investment is the gaschromatograph to start building up the smell catalogues. Furthermore, an intensive should be created to the declining hardware companies to develop portable Stinky smell generators. The budget allocated should be sufficient for a small scale experiment within our premises and two rural areas (to avoid contamination and smell interference).

Aside, a modest investment into chemical industry focussed at artifical odors may well recover the budget to conquer the perfume market.

The budget for software and research covers the cost to create the Stinky interoperable platform.

To make the product a success, it is mandatory to setup an experienced end-user group. The envisioned team should consists of two bio-chemists (one experienced in biogen manipulation), an experienced perfume composer, 3 world travellers with substantial smell experience, and a clochard from France.

The mile-stones for the MicroSmell product division are modest and confine the risks. The unpredictable risk is the effect of our product line on politics and legislation. It might be prudent to start a lobbying campaign to highlight the National health benefits from using virtual relations, such as sexual intercourse and working in contaminated areas. This might avoid the effects currently visible in the cigarette industry, where the addiction to virtual senses leads to severe marketing problems.

Research agenda and architecture

A successfull launch of the product requires innovative research in several areas. Fortunately, most of the ground work has already be done over the last two decades and the presentations of the Dagstuhl Workshop indicates that some solutions are around the corner. In particular, we need some additional work in the following areas:

- *Stinky* metrics for classification and clustering.
- Culturally biased *stinky* models

- Compression of odor functions into *stinky* physical representation scheme
- Efficient index structures for *stinky* tables
- *Stinky* algebra and smell rewrite rules
- *Stinky* cache replacement and coherency policy

The MicroSmell communicates through the *Stinky* Query Language (SQL) interface. Standardization of its features is well underway. We expect formal approvement to align with the introduction of SQL3, which provides us with a competitive edge in publicity.

The queries are optimized using the *Stinky* query optimizer, which combines all rotten tricks developed in query optimization over the last decade. This way it also takes care of the results published on fragments of the problem space. In particular, left-right, bushy, meadow, and marsh join queries are handled in constant time using hyper-randomized search strategies. Path expressions are taken care of by the World travellers, who will develop a Smelly Hitchhikers' Guide through the search space.

The *Stinky* access methods use the global information super highway to access the smell catalogues maintained by the front-offices of MicroSmell International. The results obtained for indexing and caching should provide a uniform access to every place on earth.

The *Stinky* storage system is based on both digitized representations of the key odor structures. Moreover, we will install a series of MicroSmell collectors at strategic locations to provide up to date information on the relevant odors. A few portable MicroSmells are envisioned to monitor odor emission at key happenings, such as the nomination of the next president of the US.

Exploitation and future plans

The exploitation of the MicroSmell and project results should be taken up by marketing and public twist departments. To enhance the marketing campaign stinky slogans need to be developed. Some possible slogans are: "Stinky has smell and taste" or "Stinky is a What You See Is What You Smell (WYSIWYS) device".

The *Stinky* project will further focus on smell tertiary storage structures so as to enable time-travel trips in the next century to all environmental disasters of the recent past.

A side-track activity will be to set up an Internet Sniffer and Broker to capture the results of the many hobbyists in producing smells with high marketing potential. Moreover, it provides the framework to collect the base data for customer profiling using the data mining technology provided by Data Distilleries (Amsterdam, The Netherlands).

To secure the future developments we expect the ISO-4711 standard to be approved before the turn of the century.

Last, but not least, the *Stinky* project generates a number of challenging research items that will keep its members occupied through their retirement (and perhaps after their death). Topics of interest are: "Synchronization of MicroSmells in a crowded room", "Adaptive Smell compensation algorithms", and "Smell-and-Taste of *real* multi-media WYSIWYT interfaces".