# Experiences Building the Open OODB Query Optimizer

**By Jose A. Blakeley, Texas Instruments**

Presented By **Pradeep Jagannath**

March 4, 2002

## Roadmap

## Early efforts in Query Optimization

- Object Algebra
- Query Rewriting Techniques
- Indexing Techniques
- New Execution algorithms
  - Efficient traversal of Complex Object Structures
  - Complex Object Assembly

*However, little has been reported on the development of complete working object query optimizers !*

## Major Goals in Designing Open OODB

- Extensibility
  1. new algebraic operators
  2. new algebraic transformation rules
  3. new execution algorithms
  4. improved statistics and cost models
  5. physical formats and structures
  6. enforcer algorithms for physical properties(e.g. sort order)
  7. new state space search stratergies, and
  8. improved quality of plans(e.g. thoroughness of search)
- Cost-effective rapid development
- Performance
  - Moderately complex queries should be optimized in less than 1 sec.
- Effectiveness

## Contributions of this Paper

- Design and development of a complete OODB optimization framework
- Reexamining the algenra for specifying queries on whhich the optimizers work
- Suggesting a new operator materialize for bring objects connected through path-expressions in to scope
- Revisioning the importance of physical properties in the search process
- Suggesting a new physical property *presence in memory*
- Experimental evidence to show that techniques designed in the relational context are applicable in context of OODB also
- Validation of the volcano optimizer generator for the generation of an object query optimizer

## Previous works

- Mitchell et al and Sciore and Sieg
  *Optimizer design consists of a collection of optimization regions, each of which can transform queries according to a particular control startegy,a set of algebraic transformations and a cost model. A global optimizer control coordinates the movement of query among these regions*
- Orenstein et al
  *Dynamic plan selection capability*
  However, this was not yet validated in an implementation !
  Drawbacks:
  - Uses only index based scans
  - It is not cost based
  - Optimizer was based on query algebra

## Previous works (Contd.)

- Cluet and Delobel
  - Use type information to decompose initial complex arguments into a set of simpler operators.
  - Prune the search space based on physical information(clustering, indexing).
  - Common subexpressions factorization is an important issue in OODB
- Straube and Ozsu
  A query processing methodology that includes a formal oject calculus and algebra.

## Open OODB : User Query Language

- designed to be well-integrated with C++
- uses the C++ object data model

Example:
Set ¡Newobject¿ *result;
Date lr(01,01,2002); result = SELECT Newobject(e.name(), d.name())
FROM Employee e IN Employees, Department d IN Departments
WHERE d.floor==3 and e.age()¿=32 and e.department()==d;

## Open OODB : Logical Algebra

Basic operations:

- Operators borrowed from relational domain
  *select, project, join, intersection and union*
- *unnest* — used to manipulate set–valued components
- *materialize or mat* — used to indicate the use of inter-object refeerences explicitly

## Query Simplification

This is used to transform ZOL[C++] parse trees into an equivalent algebraic operator graph

Example:

Project e.name(), d.name()
—
SELECT d.floor==3 and e.age()¿=32 and e.department()==d Get
Employees e —
Get Departments d

## Execution Algorithm

- Set processing algorithms — file and index-scan
- Value based matching — intersection, union and join
- Assembly generation

## Properties and Property Enforcement

- logical properties
- physical properties
  E.g.: *presence in memory*

The execution algorithms implement a logical operator ‘enforce’ which forces a certain physical property on the operators

## Cost model

Currently only CPU and I/O costs are considered

- sequential I/O is charged less when compared to random I/O
- assembly performance costs are expected to be studied on operational systems

## Transformation and Implementation Rules

### Transformation Rules

- Since the logical algebra is based on relational constructs the transformation rules remain the same as in relational case.
- Materialize operator introduces new operations

### Implementation Rules

This establishes a correspondence between logical algebraic expressions and execution algorithms. The rules are based on the algorithms ability to deliver the logical expression with the desired physical properties and cost estimations