The Volcano Optimizer Generator: Extensibility and Efficient Search

- Prithvi Lakshminarayanan
- 301313262

Authors

- Goetz Graefe, Portland State University
  - Won the “Most Influential Paper” award in 1993
  - Worked at HP, Microsoft and currently at Google

- William J. McKenna, University of Colorado at Boulder

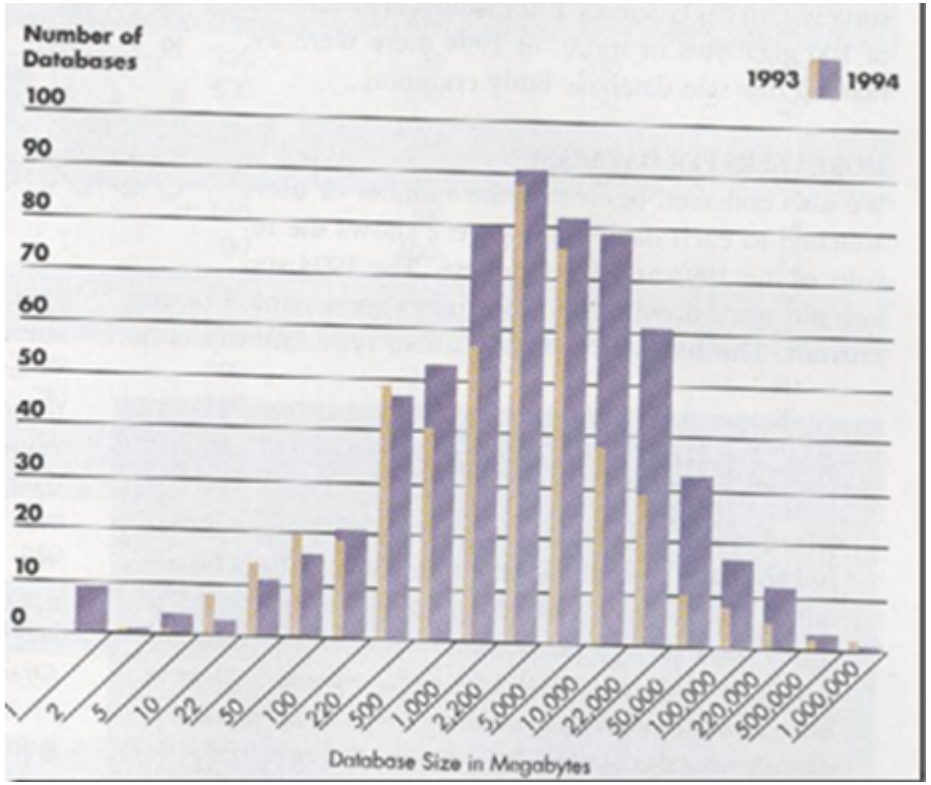
Flashback to 1993

Developments in 1993

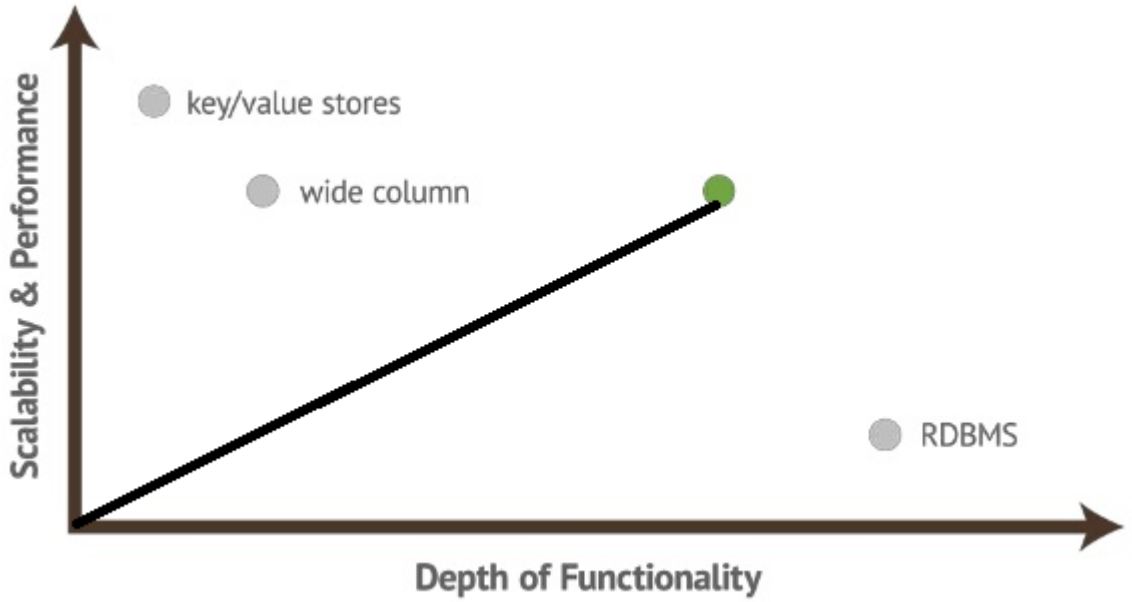
1993: Governments join in on the fun
In 1993, both the White House and the United Nations came online, marking the beginning of the **.gov** and **.org** domain names.

ODBMS

Oracle`s annual report 1993-94



Problem

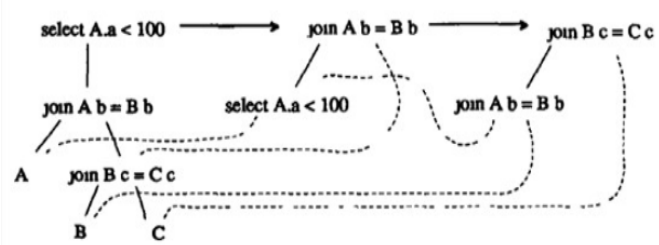

key/value stores
wide column
RDBMS
Scalability & Performance
Depth of Functionality

Problem

- Increase in data volumes
- Acceptance problems in emerging database applications
- Optimization- Speed in which the data is being retrieved
- Parallelization- Improving performance
- Improved optimizer generator over EXODUS

EXODUS

- Compiles an optimizer based on a set of rules
- Every expressions are stored in a “MESH”
- Pros:
  - Very extensible architecture
- Cons:
  - Logical and Physical operators in the same “MESH”
  - Cost ineffective



Ref paper: “The EXODUS optimizer generator”, G.Graefe, 1993

Logical and Physical operators

- Logical Operators:
  - Logical operators test for the truth of some condition
  - Returns a Boolean data type with a value of TRUE, FALSE, or UNKNOWN
  - Example- ALL, AND, ANY
- Physical Operators:
  - Physical operators implement the operation described by logical operators.
  - Each physical operators perform an operation
  - Example- INIT(), GETNEXT(),CLOSE()

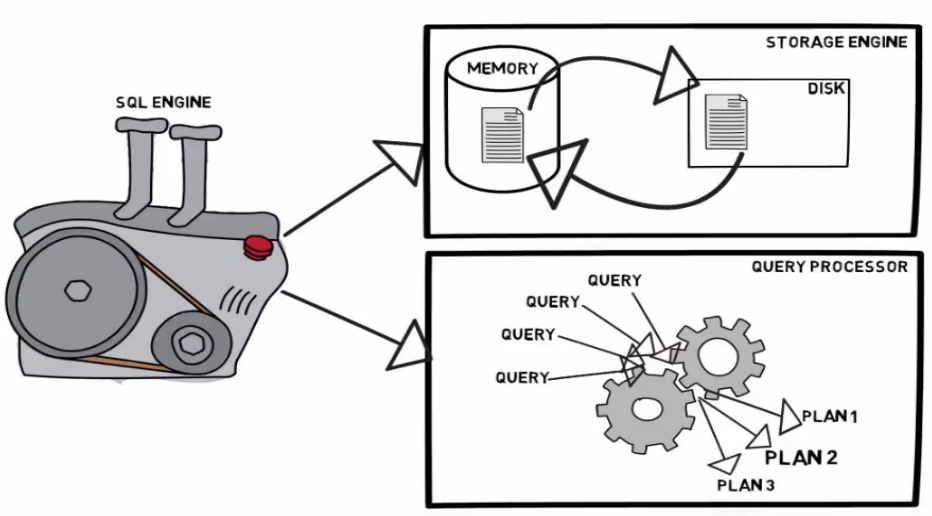
It has to…

- Act as a stand alone
- Be more efficient in time and memory consumption
- Provide effective, efficient and extensible support
- Permit the use of data models and heuristics
- Flexible cost models

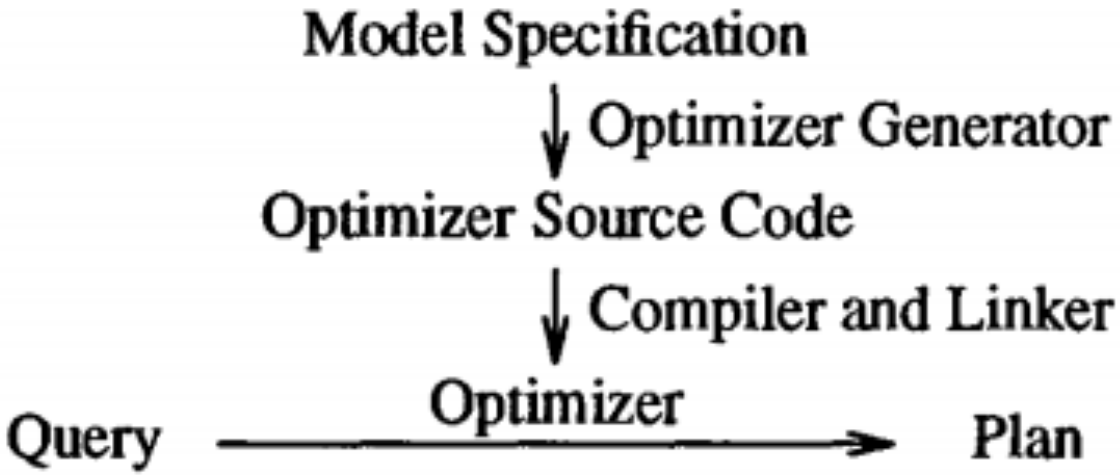
VOLCANO

What is Query Optimization?

- Optimizer generators take in a model of queries and plans and output a query optimizer that performs the actual optimization.
- Query optimizer attempts to determine the most efficient way to execute a given query
- It produces an execution plan and code generator to execute



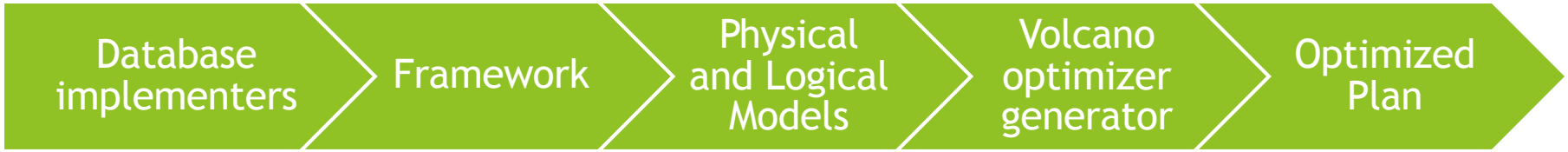
The Generator Paradigm



Optimizer Implementer – Person who specifies the data model and implements the DBMS software

DBMS User- Person poses queries to be optimized and executed

Volcano optimizer at a glance



Design Principles

- Relational algebra being used to support Object Oriented Systems
  - Logical algebra – Query
  - Physical algebra- Query evaluation plan which has the algorithms
- Creating ‘Rules’ to ensure modularity
  - knowledge about patterns in a concise and modular fashion
  - knowledge of algebraic laws as required for equivalence transformations
- Input as algebraic equivalences
- Rule compilation over interpretation due to faster execution
- Dynamic programming

Optimizer Generator Input and Optimizer Operation

- User queries as algebraic expressions
- Optimization is mapping a logical algebra expression into the optimal equivalent physical algebra expression
- Implementation algorithm is chosen
- Transformation and implementation rules for complex mappings
- Generates a logical expression and a physical property vector
- Cost function to measure CPU time, total elapsed time

Steps in Optimizer Generator Input and Optimizer Operation

- User queries
- The optimizer reorders operators and selects implementation algorithms
- Check if the rules get satisfied
- Cost function gets evoked
- Optimizer generator gets built

Search Algorithm

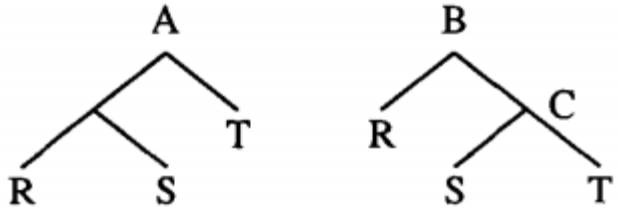
```
FindBestPlan (LogExpr, PhysProp, Limit)
if LogExpr & PhysProp is in the hash table
      if cost in hash table < Limit
            return plan and cost
      else
            return failure
else /* optimization required */
      create the set of possible "moves" from
            applicable transformations
            algorithms that give the required PhysProp
            enforcers for required PhysProp
      order the set of moves by promise
      for the most promising moves
            if the move uses a transformation
                  apply the transformation creating NewLogExpr
                  call FindBestPlan (NewLogExpr, PhysProp, Limit)
            else if the move uses an algorithm
                  TotalCost := cost of the algorithm
                  for all inputs I while TotalCost < Limit
                        determine required phys. prop. for I
                        find cost by calling FindBestPlan
                        add cost to TotalCost
            else /* move uses an enforcer */
                  TotalCost := cost of the enforcer
                  modify PhysProp for enforced property
                  call FindBestPlan for LogExpr w/ modified PhysProp
      /* maintain hash table of explored facts */
      if LogExpr is not in hash table
            insert LogExpr into hash table
      insert PhysProp and best plan found into hash table
```

Transformed using a Transformation rule
Algorithms can deliver Logical expressions with Physical properties
Enforcer might be useful to permit additional algorithm choices

The Search Engine

- Search engine can be used in all optimizers
- Dynamic programming and very goal oriented than EXODUS
- “Backward chaining”- explores only the subqueries and plans that truly participate in a larger expression
- A Hash table is used to prevent redundant optimization

Associativity Rule



- A and B are equivalent
- Expression C is not equivalent to any expression on the left
- Hence, a new equivalence is created and optimized
- This is done for the cost effectiveness and optimization in of expression B

Comparing with EXODUS

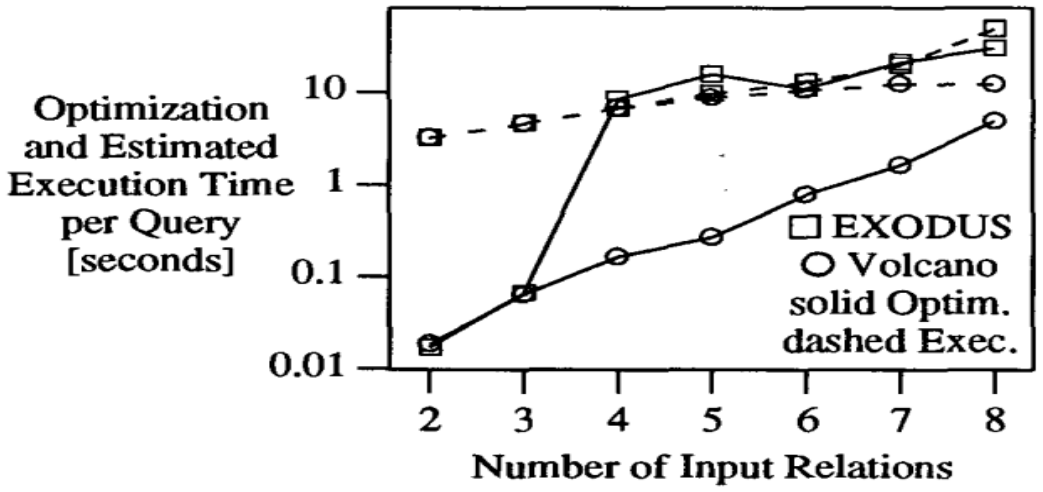
- The EXODUS code was messy and slow
- EXODUS did not distinguish logical from physical algebra which led to some inefficiencies
- EXODUS had no notion of physical properties
- EXODUS did not have a generic cost function
- EXODUS did not have the ability to modify search strategy like volcano
- Volcano took less time to optimize

EXODUS Vs Volcano

EXODUS	Volcano
Works on a MESH which had logical and physical expressions in them	Clearly able to distinguish logical and physical expressions
Physical and logical properties worked **bad** together	Physical and logical properties worked **good** together
Bottom-up approach	Top-down approach; sub expressions are optimized only if wanted
Cost is not well defined	Cost is well defined for optimizer implementation
Cannot be made extensible	More extensible

Average optimization effort

- Solid line Sun SparcStation-1 with 12 MIPS
- Dashed line indicates the estimated plan execution time
- X-axis is the 8 binary joins and Y- axis is the logarithmic



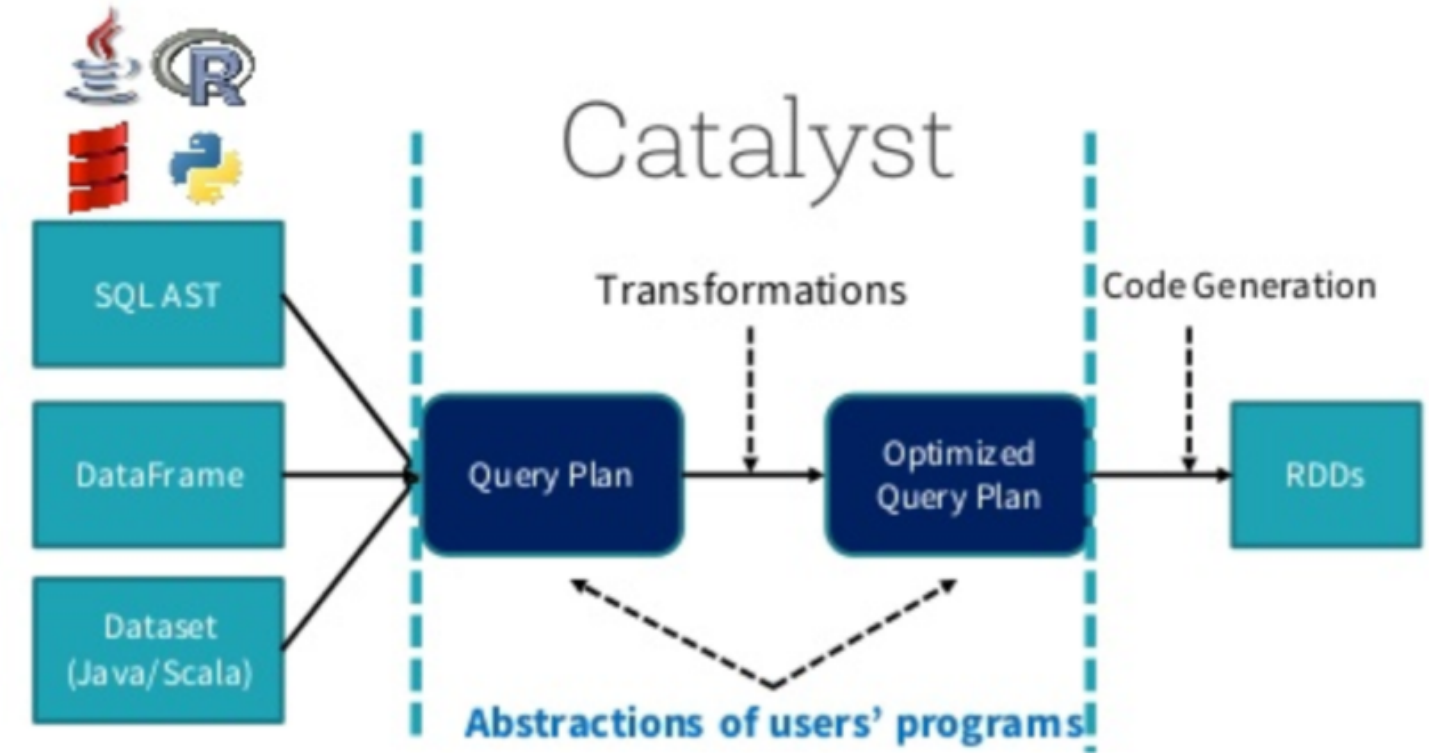
Results and findings

- The Volcano-generated optimizer performed exhaustive search for all queries with less than 1 MB of work space
- The increase of Volcano's optimization costs is about exponential
- The Volcano optimizer generator is not only more extensible, it is also much more efficient and effective than the earlier EXODUS

Spark SQL`s Catalyst

- Spark SQL is one of the newest and most technically involved components
- Catalyst optimizer leverages advanced programming language features
- Helpful in advanced analytics and for external developers
- It offers perform analysis, optimization, planning, and runtime code generation.

Spark SQL`s Catalyst



Summary

- Satisfies higher functionality and performance by having efficient tools for query processing
- Query processing engine independent of any data model
- Algebraic equivalence rules are suitable for query optimization
- Volcano optimizers are better than EXODUS

Questions from me!

- Was the performance and functionality tradeoff satisfied in Volcano?
- The paper does not speak about the drawbacks of volcano when compared with EXODUS
- What are the cases when the optimization fails in Volcano?
- What next after volcano?

Questions for me?