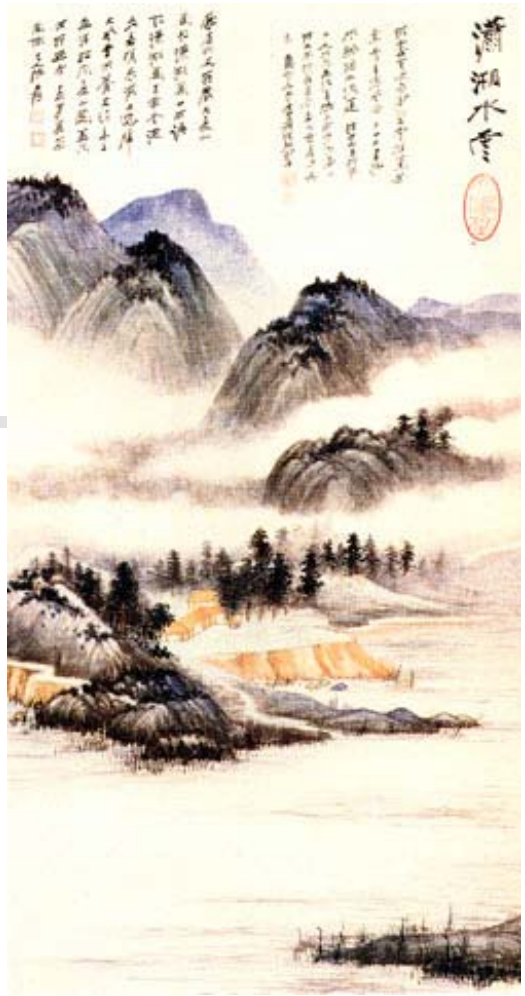
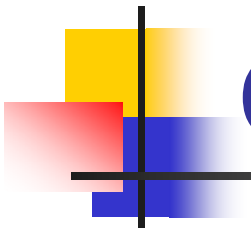
# The Volcano Optimizer Generator

*Jan. 29 2003*

Presented by Peng Wang
in CIS650

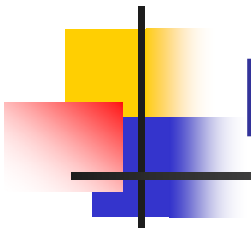


*Cite: Zhang DaQian Chinese Painting "XiaoXiangShuiYun"*



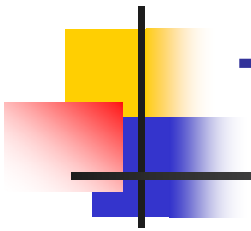
# Outline

- Background Introduction
- The Motivation: extensibility and performance
- Optimizer Generator & Search Engine
- Comparison, Summary and Evaluation



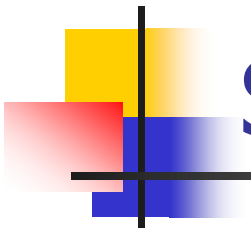
# Background Introduction

- Variations on Relational Database
  - Mid-90s: spatial DBs, OO DBs, Active DBs, etc.
- Optimizer Generator System
  - General "toolkits" for creating customized DBs
  - Exodus (Graefe&DeWitt,87)
  - Volcano (Graefe&McKenna,93)
  - Extensible and Effective
- Exodus
  - Rule based, non-exhaustive search algorithm
  - MESH structure



# The Motivation of Volcano

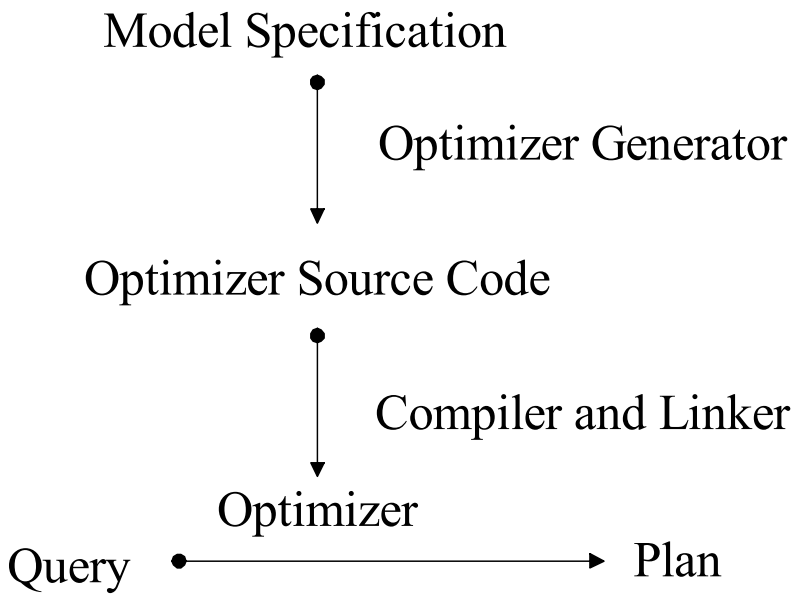
- High Performance
  - Optimization time
  - Memory consumption for search
- More Extensibility
  - Optimization rules
  - Support for physical properties
- Flexibility
  - Independent tools for optimization
  - Flexible cost modes support



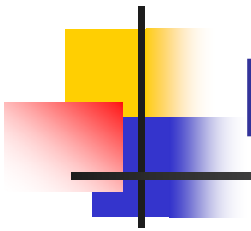
# Solution

- Rule base optimization
- Logical and physical property separation
- Directed dynamic programming
- Branch and Bound heuristic search

# Outside View of Optimizer Generator




The Generator Paradigm

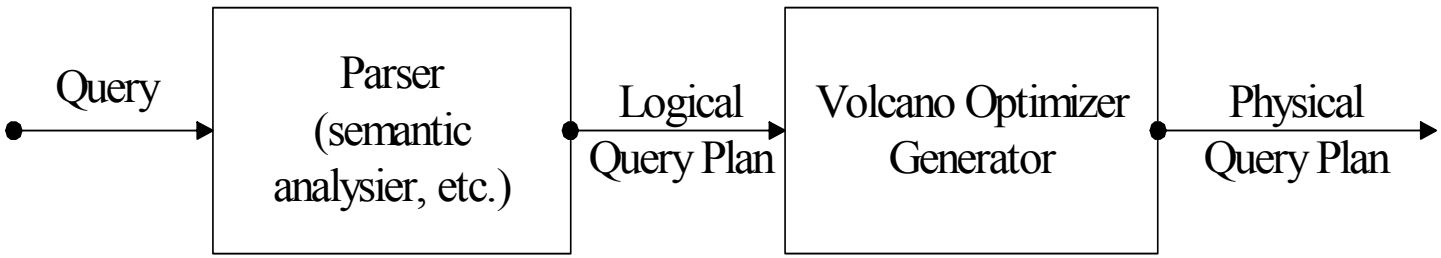


# Design Principles

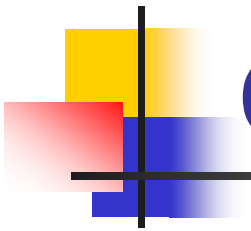
- Relational algebra
  - The technique for query processing in both relation system and extensible, object-oriented system
  - Logical and physical algebra
- Patterns and rules
  - Independent rules, modularization
- Optimal query evaluation mapping
  - Algebraic equivalence, more convenient
- Rule interpretation VS compilation
- Dynamic programming

# Optimizer Generator

- What's the input and output of the Volcano optimizer generator?

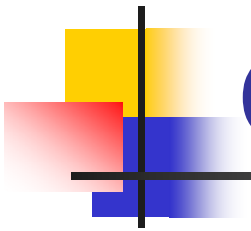



- What's the rules?
  - Rule: take from logical expression to physical expression, as well as the alternative logical expression



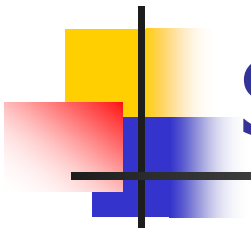
# Optimizer Generator (cont.)

- Rules:
  - If c(x), logical expression -> physical expression
  - If c(x), logical expression -> logical expression
- Examples:
  - Join(A,B) -> Hash Join(A,B) (unsorted, cost=…)
  - Join(A,B) -> Join(B,A)



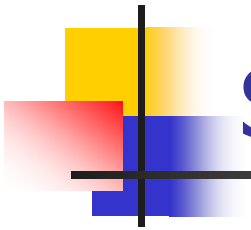
# Optimizer Generator (cont.)

- Why rules and what they need to do?
  - Extensibility
  - Algebraic transformation rules
  - Implementation rules
- Why logical and physical properties?
  - Logical properties: equivalent logical expressions
  - Physical properties: specific plans and algorithm
  - MESH in Exodus



# Search Engine

- Central component of query optimizer
- Generic but customizable search engine
  - Why generic but customizable?
- Directed dynamic programming
  - FindBestPlan procedure

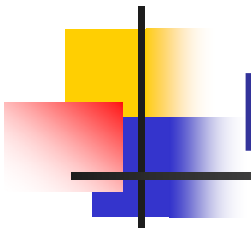


# Search Algorithm

```
FindBestPlan(LogExpr, PhyProp, Limit)
If the pair LogExpr and PhysProp is in nthe look-up table
  if the cost in the look-up table <Limit
     return Plan and Cost
  else
    return failure
/*else: optimization required*/
Create the set of possible "move" 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)
```

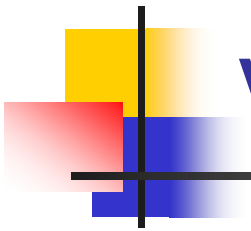
# Search Algorithm (cont.)

```
else if the move uses an algorithm
   TotalCost := cost of the algorithm
   for each input I while TotalCost <= Limit
      determine required physical properties PP for I
      Cost = FindBestPlan(I,PP,Limit – TotalCost)
      add Cost to TotalCost
  else /*move uses an enforcer*/
   TotalCost := cost of the enforcer
   modify PhysProp for enforced property
  call FindBestPlan for LogExpr with new PhysProp
/*Maintain the look-up table of explored facts*/
If LogExpr is not in the look-up table
   insert LogExpr into the look-up table
Insert PhysProp and best plan found into look-up table
Return best Plan and Cost
```



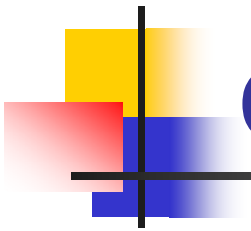
# Property of the search algorithm

- Optimal sub-plan
- Branch-and-bound pruning
  - Cost limit as bound
  - Optimization speed
- Goal-oriented
  - Only necessary part, not feasible
  - Backward chaining



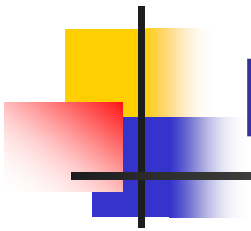
# Volcano VS Exodus

- Functionality and Extensibility
  - Logical, physical expression VS MESH
  - Top-down driven VS Always followed
  - Cost definition, extensibility
- Search Efficiency and Effectiveness
  - Small “data model”



# Contributions of Volcano

- A new optimizer generator
- Heuristic transformations VS. Cost-Sensitive
- Physical and logical property separation
- Extensibility and efficiency



# Evaluation and Discussion

- What's the weakness of Volcano?
  - Possible overlap of rules?
  - How to do "goal" directed?
- Any Questions?

# Thank you!

*Shi Tao (1642-1707) Chinese Painting*