An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment

by Andreas Winter

based on work of Chuan Zhang, Xin Yao,
Senior Member, *IEEE*, and Jian Yang

Structure

I Introduction

- data warehouse
- materialized views
- algorithms

II Materialized view selection

- query optimization
- multiple query optimization

III Algorithms for materialized view selection

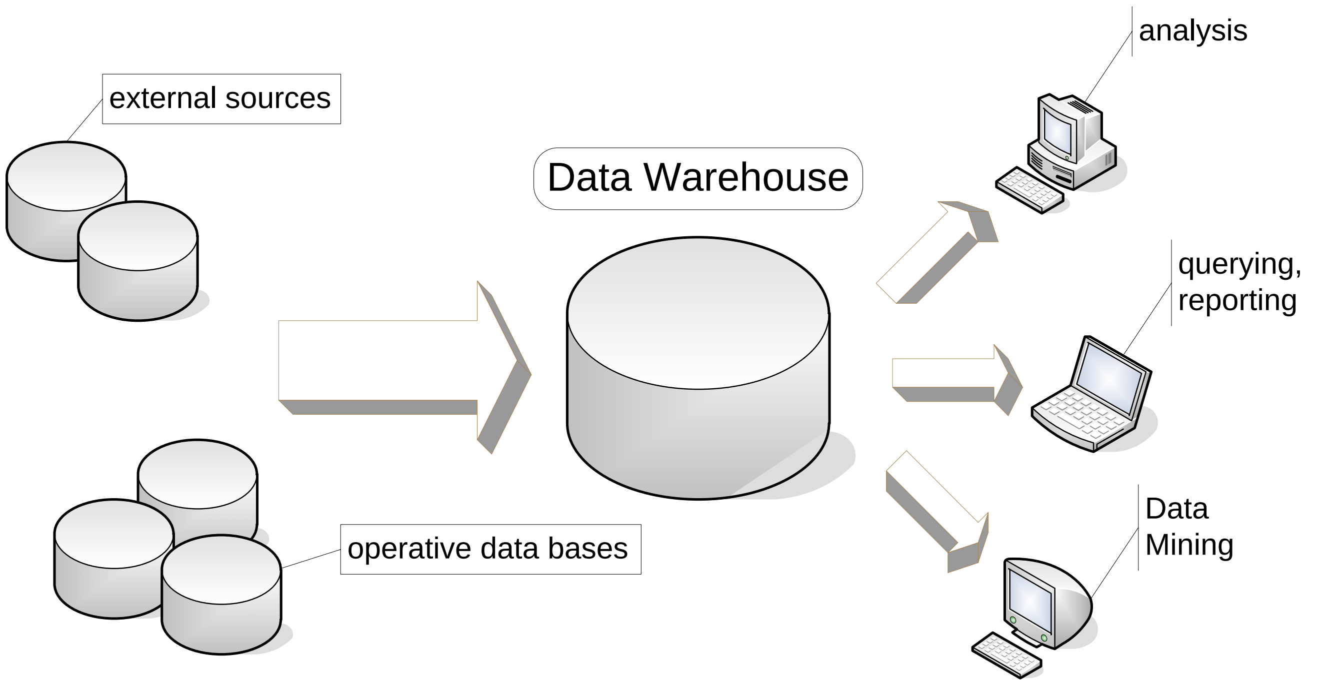
- 2-Level framework
- representation of solutions

IV Experimental Studies

V Conclusion

I. *Introduction*
Data Warehouse

- simplified view



I. *Introduction*
Materialized views

<u>problem:</u>

" What views should be materialized in order to make the sum of the query performance and view maintenance cost minimal? "

- selection involves difficult trade-off
  - materialized all views - best performance, but highest cost of view maintenance
  - materialized no views - lowest view maintenance, but poorest query performance
  - some materialized views - near optimal balance

I. *Introduction*
Algorithms

(1) deterministic algorithms
- construct or search solution in deterministic manner
- by apply heuristics or exhaustive search

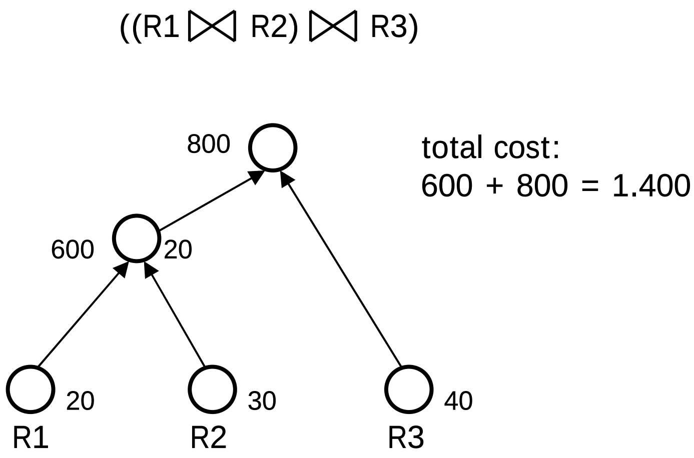
(2) randomized algorithms
- moves constitute edges between different solution
- transforming by exactly one move, solutions are connected
- each algorithm performs random walk
- no more applicable ones exists or time limit exceeded, algorithm terminate

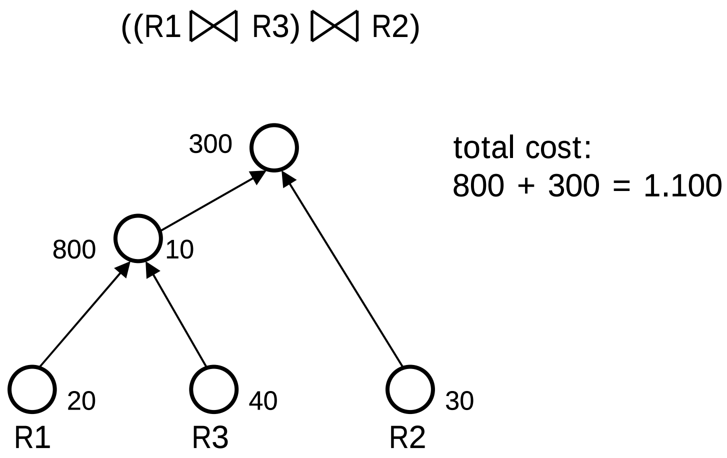
(3) evolutionary algorithms
- randomized search strategy similar biological evolution
- fittest members survive the selection

(4) hybrid algorithms
- combine deterministic, randomized and evolutionary algorithms
- e.g. deterministic algorithms solutions can be used as starting points for randomized algorithms

II. *Materialized view selection*
Query optimization

- join operation is one of the most expensive operations
- for example: R1 = 20, R2 = 30, R3 = 40

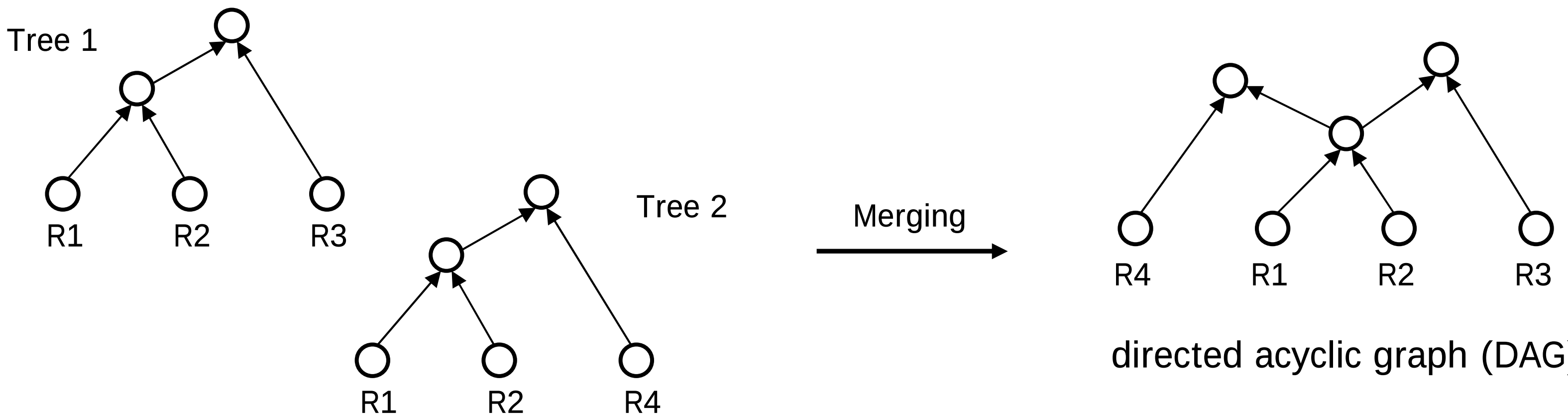




- ***goal*** : find a processing plan with lowest query processing cost

II. *Materialized view selection*
Multiple query optimization

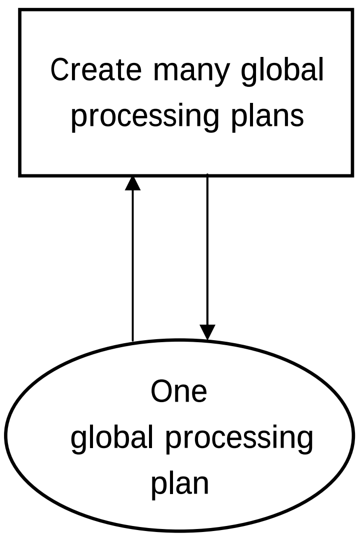
- ***goal*** : find a global/multiple processing plan such the query cost is minimized



- in general, union of locally optimal plans ≠ globally optimal plan
- algorithm is often needed

III. *Algorithms for materialized view selection*
2-Level-Framework

- algorithms based on the 2-level structure



higher level
(global processing plan optimization)

lower level
(Materialized view selection based on one global processing plan)

III. *Algorithms for materialized view selection*
Representation of global processing plans

- higher level optimization
- queries Q_1, Q_2 ... Q_n
- global processing plan represented by a vector of *n* integers

  {[P_{1i}], [P_{2j}], ... [P_{kn}]} P_{kn} .. *k*th local processing plan for Q_n

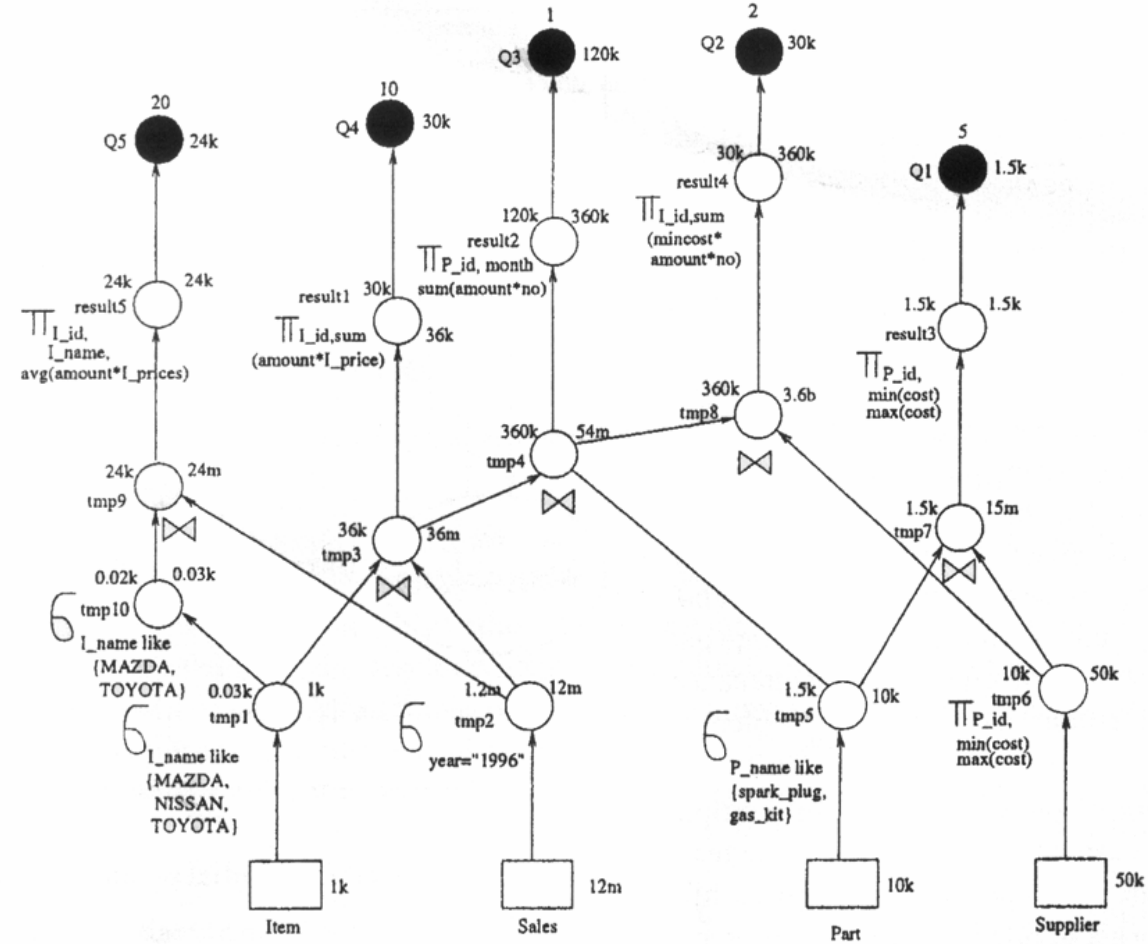
- for example:
  - number of local processing plans for Q_1 = 12, Q_2 = 120, Q_3 = 80
  - vector {[4], [89], [70]} reprents a global processing plan, that means 4th processing plan for Q_1, 89th for Q_2 and 70th for Q_3
  - range for each plan is [1 ... 12], [1 ... 120] and [1 ... 80]

III. *Algorithms for materialized view selection*
Representation of materialized views

- lower level optimization
- based on DAGs (directed acyclic graph)
- each DAG encoded as a binary string
- 1 indicates that the corresponding node is materialized, 0 it is not
- binary string called also mapping array
- for example:
  - breadth-first travers of the DAG results follow ordered list: {[Q5,0], [Q4,0], [Q3,0] ... [tmp6,0]}
  - binary string {0,0,0,0,0,0,0,....,0} means that no node is materialized
  - {0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1} means that nodes {Q4, Q1, result 5, tmp2, tmp5 and tmp6 } are materialized, others not

III. *Algorithms for materialized view selection*
Example

- four relations
  - Item, Part,
  - Supplier, Sales
- five queries



III. *Algorithms for materialized view selection*
Crossover

- encourages information exchange among different individuals
- assembling better individuals
- one-point crossover

- for example:

(1) lower level

crossover point = 7

individuals	L1 = 1 100 100|0 100 100 001 111	
	L2 = 0 100 110|1 011 000 100 111	
offsprings	L1‘ = 1 100 100|1 011 000 100 111	
	L2‘ = 0 100 110|0 100 100 001 111	

(2) higher level

crossover point = 3

individuals	L1 = [4][20][30][10][99]
	L2 = [5][30][21][40][80]
offsprings	L1‘ = [4][20][30][40][80]
	L2‘ = [5][30][21][10][99]

III. *Algorithms for materialized view selection*
Mutation

- needed to create new genes
- enables the algorithm to reach all possible solutions (in theory)
- for example:

(1) lower level

generate position = 16

individuals L = 11 001 000 100 100 001 111

offsprings L' = 11 001 000 100 100 011 111

(2) higher level

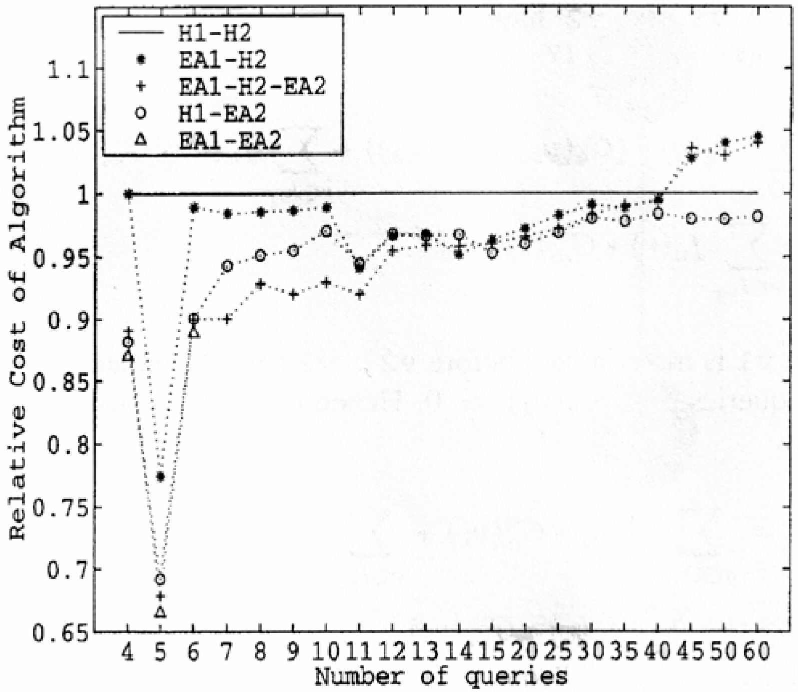
generate gene = 3

individuals L = [4][20][30][10][99]

offsprings L' = [4][20][16][10][99]

IV. *Experimental Studies*

- simulation software based on the Simple Genetic Algorithm and GAlib



COMPARISON OF TIME TO FIND THE GLOBAL OPTIMAL SOLUTION BY DIFFERENT ALGORITHMS

Algorithm	6 queries	7 queries	8 queries
H1-H2	5 Secs	50 Secs	1.2 Mins
EA1-H2	30 Secs	10.1 Mins	20 Mins
EA1-H2-GA2	1 Mins	10 Mins	25 Mins
H1-EA2	35 Secs	10 Mins	22 Mins
EA1-EA2	4 Mins	2 Hours	7 Hours
Exhaustive	5 Mins	2.5 Hours	25.2 Hours

EA1 higher level evolutionary algorithm
EA2 lower level evolutionary algorithm

H1 higher level heuristic algorithm
H2 lower level heuristic algorithm

V. *Conclusion*

- materialized view selection based on multiple query processing plans
- proposed a 2-level structure
- pure evolutionary algorithms impractical due to their excessive computation time
- pure heuristic algorithms unsatisfactory in terms of the quality of the solutions
- performance of hybrid algorithms that combine advantages of heuristic and evolutionary seems the best

"Finding the suitable trade-off between the computation time and the cost saving will be a topic for future studies."