Solving View Selection Problem To Query Plan Using Meta Heuristic Methods.

Dr.Ahmed Khalaf Zager Al saedi

Faculty of Science/ deartment of physics

Missan University

ahmedkhalafzager@uomisan.edu.iq

Abstract View Selection Problem to materialize is one of the very important decisions in designing a data Database. Decision support systems usually use complex queries on large databases. Because responding time should be small, therefore query plan is very important. In this paper, we submit four advances in view materialization. First, a more robust optimization function, Minimum View Selection Problem, and data cube lattices as well as multi-view processing plan (MVPP) are formalized. The decreased View Selection Problem allows for multiple querying nodes, partial and full materialization, and data constriction. The contribution of the research is to select an appropriate set of views that minimize total query response time and the cost of selected views. The traditional method ineffective, using Polynomial Greedy Algorithm (PGA) and genetic algorithm (GA), views of the Data Warehouse can be optimized

*.Keyword: Materialize view, Query Plan, View selection Problem, Data warehouse, Greedy algorithm, Genetic algorithm*

# Introduction

The problem of view selection based query processing plays an important role in many database applications, including data warehouse, data mining, and query optimization and information integration. The View Selection Problem is an NP-Complete problem [1] dealing with the selection of views to materialize in order to eﬃciently answer queries posed to a data warehouse [2, 3, 4, 5]. View materialization is a well-documented and established technique for data warehouse optimization [6]. materialize that can answer the given set of workload queries and is optimal in some sense. Some recent researches used heuristic approaches to solve view selection problem for materialization view in data warehouse such as greedy algorithm proposed by Harinarayan, Raja Raman and Ullman (HRU) [7].who formalized this technique by modeling benefit and dependencies among queries with a weighted partial order, and The Polynomial Greedy Algorithm (PGA) proposed by Nadeau and Teorey [8], offer a more scalable alternative to HRU. However, they did not provide better solutions in terms of chosen optimization queries to reduce query execution time by decreasing number of viewers which have been selected to be materialized. High efficiency is important as used Graphical Representations such as the cube lattice and multi-view processing plant (MVPP) for queries in data warehouse to aid algorithms in solving the view selection problem (VSP). The objective of this paper is to propose three advances. First, using greedy algorithm for selecting materialized views introduced by Harinarayan as HRU, calculating the benefit of each possible view during each iteration, and selecting the most beneficial view for materialization based on cube lattice. Second, comparing respond time query for HRU [7] with Nadeau and Teorey [8] using Polynomial Greedy Algorithm (PGA). Third, using genetic algorithm to solve view selection problem on multi-view processing plant (MVPP) to minimize the query cost. After the experiment, it was discovered that the Polynomial Greedy Algorithm (PGA) offers a more scalable alternative to HRU and faster query responding time. On the other hand, also genetic algorithm give us lower query cost. View materialization is a well-documented and established technique for data warehouse optimization..

# Query plan

When a query is posed to an OLAP system, there may be multiple materialized views available that could be used to compute the result. For example, if we have the situation represented in Fig. 1. and a user Method a query plan to set rows by day and state, that query is naturally answered from the view labeled (1, 2). Ho while sever, since (1,1) is not materialized, our study need to find a materialized ancestor to extracting and acess the data. There are three such nodes in the product graph of Fig. 1. The query can be answered from nodes (0, 0), (1, 0), or (0, 2). in the chance of answering queries plans from substitutional sources, the enhancement problems emerge upon which source is the greatest active for the set query plan. Most existing research focuses on syntactic approaches. The contingent query version are executed, alternative query costs are estimated, and what show to be the better or promising plan is completed, alternative query costs are estimated, and what show to be the better or promising plan is completed. Another approach is to query a metadata table containing information on the materialized views to determine the best view to query against, and then translate the original SQL query to use the best view.



Fig.1. Product graph labeled with aggregation level coordinates

# A HEURISTIC APPROACH FOR SELECTING MATERIALIZED VIEW.

 A number of heuristics have been presented in this paper to solve the View Select Problem (VSP). In this Fig. 2. Framework below shows the extract sty of query plans and attribute of the query, to generate the set of candidate views to reduction the time and cost.



Fig.2. Framework of the study

## Greedy Algorithm

A greedy algorithm chooses, at each step, the locally optimal move in hopes of ultimately ﬁnding the global optimum. From inception to early adoption, the VSP was solved exclusively by greedy heuristics. In their seminal article on the View Selection Problem, Harinarayan et al [9] formalized the cube lattice and proposed two greedy heuristics: Greedy and Beneﬁt per Unit Space (BPUS). These algorithms work as follows. Greedy is employed on VSP models with a predetermined number of views to materialize k.The algorithm iterates over every view in the remaining candidate set (which is initialized to every view minus the root which must be materialized by deﬁnition), selecting for materialization the view with the greatest beneﬁt. The beneﬁt of a view v is simply deﬁned as the current objective function value given a set of materialized views M minus the objective function value including v in M, i.e., OF (M) − OF (M∪{v}). The process continues until k views are selected.

In this paper the space cost and time are calculated according the number of rows in the view. Let S (v) be the cost to view selectin v. The set of views to be materialized should be saved and always include the top view to answer the query corresponding to the view. Suppose there is limit K on the number of views that we may select after selecting a few sets of S of views, the benefit of view v relative to S denoted by B (v, S) is defined as follows.

For each W ⊆ V define the quantity Bw by:

Let U be the view of least cost in S such that W⊆ U

If C (v) < c (u), then Bw =c (v)-c (u).

Define B(v, S) = ∑\_(W≤v)▒B\_w

Now, we can define the greedy algorithm for selecting a set of K views to materialize. The algorithm is shown in Fig. 3.



Fig .3. The greedy selection

## Cube lattice

One of the earliest generated VSP representations, and the most commonly used, is the Data Cube Lattice (simple cube lattice) [9, 10, 11,12] proposed by Harinarayan . The cube lattice is a directed acyclic



Fig .4. Example of a hypercube lattice structure [Harinarayan et al. 1996].

Parent (v) = {u | v ⊆ u; x, v ⊆ x, x ⊆ u}

Siblings (v) = {u | Level (u) = Level (v) ∧parent (u) = parent (v)} children (v) = {u |u ⊆ v; x, u ⊆ x, x ⊆ v}

Ancestors (v) = {u |v ⊆ u}

Descendants (v) = {u |u ⊆ v}

The cube lattice can contain any set of views one desires to consider for materialization. However, since the number of possible views can be quite large, the cube lattice is generally formed by queries found in the query. For the greedy algorithm to function in the cube lattice, we must make three successful choices (iterations) of view to materialize. In table 1 given the benefit of each of the view besides {c,p,s), when calculating the benefit we begin with the assumption that each view is evaluated using {c,p,s}. If we pick view {p,s} to materialize first then we reduce its cost by 5.2 and that for each of the views as shown in table 1.

1. Two Iterations of HRU, based on Fig. 4.

|  |  |  |
| --- | --- | --- |
| selection | Iteration 1 benefit | Iteration2 benefit |
| {p,s} | 5.2M X 4=20.8M | 0 |
| {c,s} | 0 x 4=0 | 0 x 2=0 |
| {c,p} | 0x4=0 | 0 x 2=0 |
| {s} | 5.99M 2=11.98M | 0.79Mx2= .58M |
| {p} | 5.8 M X2 =11.6M | 0.6M X 2 =11.8M |
| {c} | 5.9M X 2=11.8M | 5.9 M X 2=11.8 |
| {} | 6M-1 | 0.8M-1 |

Table 1 shows the calculations for the first two iterations of HRU. Materializing {p, s} saves 6M – 0.8M = 5.2M rows for each of four views: {p, s} and its three descendants: {p}, {s}, and {}. The view {c, s} yields no benefit materialized, since any query that can be answered by reading 6M rows from {c, s} can also be answered by reading 6M rows from the fact table {c, p, s}. HRU calculates the benefits of each possible view materialization. The view {p, s} is selected for materialization in the first iteration. The view {c} is selected in the second iteration.

 HRU is a greedy algorithm that does not guarantee an optimal solution, although testing has shown that it usually produces a good solution. Further research has built upon HRU, accounting for the presence of index structures, update costs, and query frequencies.

HRU evaluates every unselected node during each iteration, and each evaluation considers the effect on every descendant. The algorithm consumes O (kn2) time, where k = |views to select| and n = |nodes|. This order of complexity looks very good; it is polynomial time. However, the result is misleading. The nodes of the hypercube lattice structure constitute a power set. The number of possible views is therefore 2d where d = |dimensions|. Thus, n = 2d, and the time complexity of HRU is O (k22d). HRU runs in time exponentially relative to the number of dimensions in the database.

## Polynomial Greedy Algorithm (PGA)

The Polynomial Greedy Algorithm (PGA) [10] offers a more scalable alternative to HRU. PGA, like HRU, also selects one view for materialization with each iteration. However, PGA divides each iteration into a nomination phase and a selection phase. The first phase nominates promising views into a candidate set. The second phase estimates the benefits of materializing each candidate, and selects the view with the highest evaluation for materialization.

Table2. First Iteration of PGA, Based on Fig. 3.

|  |  |
| --- | --- |
| Candidate  | Iteration 1 benefit  |
| {p ,s} | 5.2M X 4=20.8M |
| {s} | 5.99M x 2=11.98M |
| {} | 6M-1 |

The nomination phase begins at the top of the lattice; in Fig .3. This is the node {c, p, s}. PGA nominates the smallest node from amongst the children. The candidate set is now {{p, s}}. PGA then examines the children of {p, s} and nominates the smallest child, {s}. The process repeats until the bottom of the lattice is reached. The candidate set is then {{p, s}, {s}, {}}. when a track or path of selection views has been installed, the algorithm enters the selection bands. The resulting calculations are shown in Tables 2 and 3.Compare Tables 2 and 3 with Table 1. Notice PGA does fewer calculations than HRU, and yet in this example reaches the same decisions as HRU. PGA usually picks a set of views nearly as beneficial as those chosen by HRU, and yet PGA is able to function when HRU fails due to the exponential complexity. PGA is polynomial relative to the number of dimensions. When HRU fails, PGA Extends the usefulness of the OLAP system. The materialized view selection algorithms discussed so far are static; that is, the views are picked once and then materialized. An entirely different approach to the selection of materialized views is to treat the problem similar to memory management. The materialized views constitute a view pool. Metadata is tracked on usage of the views. The system monitors both space and update window constraints. The contents of the view pool are adjusted dynamically. As queries are posed, views are added appropriately. Whenever a constraint is violated, the system selects a view for eviction.

|  |  |
| --- | --- |
|  Candidate | Iteration 2 benefit |
|  {c,s} | **0 x 2=0** |
|  {s} | **0.79M x2= 1.58M** |
|  {c} | **5.9 M X 2=11.8** |
|  {} | **0.8M-1** |

 ABLE 3. SECOND ITERATION OF PGA, BASED ON FIG. 3***.***

# query execuation time (test case)

The algorithm consumes (kn) time, where k = |views to select| and n = |nodes|. This order of complexity looks very good; it is polynomial time.

Table 5 time of two iteration of RUH, Based on Fig. 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Candidate  | size  | Time(KN)for iteration 1 | Time(KN) for iteration 2 | Time ms |
| {p,s} | 0.8 m | 3.2 | 0 | 0 |
| {c,s} | 6m | 24 | 24 | 48 |
| {c,p} | 6m | 24 | 24 | 48 |
| {s} | 0.01m | 0.2 | 0.02 | 0.22 |
| {p} | 0.2m | 0.4 | 0.04 | 0.44 |
| {} | 1 | 1 | 1 | 2 |

Table 5. Time of two Iterations of PGA, Based on Fig. 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Candidate  | size  | Time(KN)for iteration 1 | Time(KN) for iteration 2 | Time ms |
| {p,s} | 0.8 m | 3.2 | 0 | 3.2 |
| {s} | 0.01 | 0.02 | 0.02 | 0.04 |
| {} | 1 | 1 | 1 | 2 |
| {c,s} | 6m | 0 | 12 | 12 |
| {s} | 0.01m | 0.02 | 0.02 | 0.04 |
| {c} | 0.1m | 0 | 0.1 | 0.1 |
| {} | 1 | 1 | 1 | 2 |



Fig 4. Time and Space versus Number of Views Selected

 By algorithms

# Multi View Processing Plan

Since 2005, the most implemented graphical representation for the VSP is the Multi-View Processing Plan (MVPP) [15, 16, 17, 18, 19,] show in Fig 4 and 5 as deﬁned by Yang. MVPP is a DAG representing a query processing strategy. The root (source) nodes are the queries themselves, the leaf (sink) nodes are the base relations, and all other intermediate nodes are selection/projection/join views that contribute to the construction of a given query in a bottom-up manner. Each query has its own processing plan as shown by Fig. 5. When all plans are generated, they are merged into a single MVPP as depicted by Fig.6. All views in the resulting MVPP are candidates for materialization.

## Genetic Algorithm for Materialized View Aid by MVPP

Since GA simulates the biological process, most of the terminology is borrowed from biology. A detailed illustration of GA terminology can be found in [20]. One of the differences between GA and other commonly used techniques is that GA operates on population of strings, not a single string. Every population is called a generation. A single solution is called a Phenotype and is represented by a single string. Solutions are presented as strings (chromosomes) which are composed of characters (genes) that can take one of several values (all).



**Fig .4**. Simple Genetic Algorithm to Select View

In initial generation, G (0), and for each generation, G (t), a new generation G (t+1) is generated. An abstract view of the algorithm is shown in Fig. 2. Each problem should have its own solutions represented as character strings by an appropriate encoding. Selection, crossover and mutation are three operators applied to successive string populations to create new populations. In other words, these three operators are applied on G (t-1) to generate G (t) as shown in Fig. 2. Choosing fitness function is important.

Fitness is used in evaluating individual G (t). In GA, the average fitness and the fitness of the best solution increases with every new generation. In order to get the best solution, many generations should be evolved. Several stopping criteria exist for the algorithm. For epitome, the procedure may be stood when all solutions in a descent are identical.

We revise our GA based on the similar of Simple GA described in [2]. With some modification on the policy of selection and fitness, we propose the following version of GA which is suitable for our problem. The objective in our cost model is stated as the minimization of the sum of query cost and maintenance cost, while the objective or fitness function of GA is naturally stated as maximization. Therefore, there should be a transformation from our cost function to the fitness function in GA. For example, the commonly used transformation in GA is as follows:

F(x) = C\_(max-) C(x) Where C(x) < C max (1)

C(x) denotes the cost function. There are a lot of ways to choose the coefficient C\_max . C\_max May taken as an input coefficient as the largest c(x) value in the current population or the largest of last K generation. The crossover operator is a way of random number generation, string copies and swapping partially good solutions in order to get a better result. For example, there are two strings from our example:

# Experimental studies

This section presents an example to give a progressive overview of several key aspects of materialized view design methodology. Suppose that the member databases contains the following relations:

Result (Sid, Name, Did), section (Did, name, city), Order (Pid, Cid, quantity, date), Customer (Cid, name, city)

Part (Tid, name, Pid, supplier).

We use the shorthand’s Pd, Div, Ord, Cust, and Pt to stand for the above relations respectively. For simplicity, we assume here that these relations are all at the same site, so we will not consider the data communication cost in the following calculation. Suppose that we have the following two frequently asked data warehouse queries

**Query 1: Select Pd.name from Pd, Div Where Div.city="LA" and Pd.Did=Div.Did**

**Query 2: Select Pt.name from Pd, Pt, Div where Div.city="LA" and**

 **Pd.Did=Div.Did and Pt.Pid=Pd.Pid**

**Query 3: Select Cust.name, Pd.name, quantity From Pd, Div, Ord, Cust Where**

 **Div.city="LA" and Pd.Did=Div.Did and Pd.Pid=Ord.Pid and**

 **Ord.Cid=Cust.Cid and date>7/1/96**

**Query 4: Select Cust.city, date From Ord, Cust Where quantity>100 and**

 **Ord.Cid=Cust.Cid**

**Fig. 5.**  Our Queries



 **Fig. 6**. Individual Query Processing Graph



Fig.7. Apply MVPP merged plan for queries

Based on the principle of minimal alphabets of GA coding, the string is essentially a binary string of ones and zeroes. The representation of our problem is a MVPP, which is a DAG rather than a binary string. If we can map the representation from DAG to a binary string, we can apply GA to our problem. The mapping strategy is shown in Fig .3.

 For example, search through the DAG in Fig. 1. using width-first , we obtain the mapping array as follows { [Q4,0], [Q3,0], [Q2,0], [Q1,0], [ [result1,0], [result2,0], [result4,0], [result3,0], [tmp9,0], [tmp3,0], [tmp4,0], [tmp8,0], [tmp7,0], [tmp10,0], [tmp1,0], [tmp2,0], [tmp6,0]}, its length is 20 excluding the 5 sources which are tables, product division, part, order and customer. Suppose the result of GA is f{0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1}, that means that the Nodes {Q4, Q1, result5, tmp2, tmp5, tmp6g} should be materialized. Using Genetic Algorithm for Materialized View Selection in Data Warehouse Environments requires the following steps:

Begin

(i) Input a MVPP represented by a DAG;

(ii) Use a certain graph search strategy such as breadth first, widths or problem-oriented searching method to

an ordered investigation over, all of the node in the HAG and output can ordered string of these growth sequence of these nodes

(iii) Based on this sequence of nodes, create a two dimensional array to store the sequence of nodes and strings of 0s and 1s. One dimension is for the sequence of nodes, another dimension is for strings of 0s and 1s. Of the strings of 0s and 1s, 0 denotes that the corresponding node in the array, indexed by the same subscript, is unmaterialized. 1 represents the corresponding node in mapping array is materialized. This array is called the mapping array.

#  A Representation of Solutions in Crossover

The crossover operator is a way of random number generation, string copies and swapping partially good solutions in order to get a better result. For example, there are two strings from our example: L1 = 1100100/0100100001111 L2 = 0100110/1011000100111 L1 means that nodes {fQ5, Q4, Q1, result4, tmp3, tmp1, tmp2, tmp5, tmp6} are materialized. L2 means that nodes {Q4, Q1, result5, result2, result3, tmp9, tmp7, tmp2, tmp5, tmp6} are materialized. Suppose k is chosen from 1 to 20 randomly. We obtain a k=7 (The symbol j represents the position of crossover applied). The results of the crossover are two new strings: L- 1 = 1100100/1011000100111 L - 2 = 0100110/0100100001111 The two new individuals, L - 1 means that nodes { Q4, Q1, result2, result3, tmp9, tmp7, tmp2, tmp5, tmp6} are materialized and L - 2 means that nodes {Q4, Q1, result5, result4, tmp3, tmp1, tmp2, tmp5, tmp6}are materialized.

# RESULT

We conducted a comprehensive performance evaluation of the query optimization using java (netbense.10) and MySQL server 2014.the result shows the genetic algorithm give good initial solution and lowest cost and high convergences when using MVPP. Deterministic search for solution using heuristics in the view selection problem decreases the solution space. But when the size of the data warehouse is very large, scalability is a big issue due to exponential complexity. Though some heuristic algorithms have been designed with reduced time complexity, they are yet to be tested on very large databases and a large number of complex queries. Evolutionary approaches like GA, determine a solution to be the ﬁttest depending on predeﬁned numbers of generations and iterations shows in Figure.8



Figure 8. Time and Space Used for Metalizing for Number of Query

# Conclusion

Our analysis focused on various search methodology and the application of heuristic to provide a practical aspect to the theoretical model. We introduced the concept of full and partial materialization, as well as their impact on performance, furthermore constructor heuristic was adapted from the VSP to handle solution validity and maximize materialization compounding ,the result show the Polynomial Greedy Algorithm (PGA) degrees query execution time compere with Harinarayan, Raja Raman and Ullman (HRU). Also Genetic algorithm produced high quality solution by decrees the cost. The VSP was extended to include multiple schema type and our Data warehouse and DBMS architecture, to the best of our knowledge. Through extensive study to solve the problem (VSP), we found that it can be resolved by using heuristic approaches based on representation graphics such as cube lattice and multi-view processing plane (MVPP) of queries in the data warehouse.

 At the same time, it can also reduce the execution time and increase profit. how to select a set of views to be materialized so that the total cost of processing a set of queries and maintaining the materialized views and query execution time are minimized. The cost model takes into consideration of not only query access frequencies and base relation update frequencies, but also query access costs and view maintenance costs. The algorithms for generating MVPPs use the techniques from single query optimization, coupled with query tree merging techniques which aim to incorporate the individual optimal query plans as much as possible in the MVPP. We were able to successfully map the optimal MVPP generation problem as an O-l integer programming problem and investigated the problem of view selection in the data cube to materialize in order to minimize query response time and reduce cost.

# Reference

1. “This paper is an extensions of work originally present in conferences 1st Annual International Conference on Information and Sciences (AiCIS)2018.
2. Venky Harinarayan, Anand Rajaraman, and Jeﬀrey D. Ullman. Implementing data cubes eﬃciently. In Proceedings of the 1996 ACM SIGMOD International Conference on Management, pages 205–216, 1996.
3. Yannis Kotidis and Nick Roussopoulos. A case for dynamic view management. ACM Trans. Database Syst., 26(4):388–423, 2001
4. Don-Lin Yang, Man-Lin Huang, and Ming-Chuan Huang. Eﬃcient utilization of materialized views in a data warehouse. Lecture Notes in Computer Science, 2336:393–404, 2002.
5. A. Bauer and W. Lehner.” On solving the view selection Problem in distributed data warehouse architectures”. In

 Conference on Scientiﬁc and Statistical Database Management, 2003. 15th International, pages 43–51, 2003.

1. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
2. Himanshu Gupta and Inderpal Singh Mumick. Selection of views to materialize in a data warehouse. IEEE Transactions on Knowledge and Data Engineering, 17(1):24–43, 2005.
3. V. Harinarayan, A. Rajaraman, and J. Ullman, “Implementing Data Cubes Efficiently,” Proceedings SIGMOD 1996, 205- 216.
4. Jeﬀrey D. Ullman and Jennifer Widom. “A First Course in Database Systems”. Pearson, Prentice Hall, 3rd edition, 2008
5. Jorge Loureiro and Orlando Belo. The M-OLAP cube selection problem: a hyper-polymorphic algorithm approach. In Proceedings of the 11th international conference on intelligent data engineering and automated learning, IDEAL’10, pages 194–201, Berlin, Heidelberg, 2010. Springer-Verlag.
6. Konstantinos Morfonios, Stratis Konakas, Yannis Ioanndis, and Nikolaos Kotsis. ROLAP implementations of the data cube. ACM Computing Surveys, 39(4): Article 12, 2007.
7. Jeﬀrey Xu Yu, Xin Yao, Chi-Hon Choi, and Gang Gou. Materialized view selection as constrained evolutionary optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 33(4):458–467, November 2003.
8. Abdelmadjid Boukra and Sadek Bouroubi. Selection of views to materialize in data warehouse: A cooperative approach. Studia Informatica Universalis, 9(2):19–37, 2001.
9. Venky Harinarayan ,Anand Rajaraman and Jeffery D.ullman .implementing data cubes efficiently .ACM SIGMOD
10. Thomas P. Nadeau and Toby J. Teorey. Achieving scalability in OLAP materialized view selection. In Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, DOLAP ’02, pages 28–34, New York, NY, USA, 2002. ACM.
11. Roozbeh Derakhshan, Bela Stantic, Othmar Korn, and Frank Dehne. Parallel simulated annealing for materialized view selection in data warehousing environments. In Proceedings of the 8th international conference on Algorithms and Architectures for Parallel Processing, ICA3PP ’08, pages 121–132, Berlin, Heidelberg, 2008. Springer-Verlag
12. Roozbeh Derakhshan, Frank Dehne, Othmar Korn, and Bela Stantic. Simulated annealing for materialized view selection in data warehousing environment. In Proceedings of the 24th IASTED international conference on Database and applications, pages 89–94, Anaheim, CA, USA, 2006. ACTA Press.
13. Jiratta Phuboon-ob. Materialized View Selection Using Two-Phase Optimization Algorithm. PhD thesis, National Institute of Development Admistration, Bangkapi, Bangkok, Thailand, 2009.
14. Noha A.R. Yousri, Khalil M. Ahmed, and Nagwa M. El-Makky. Algorithms for selecting materialized views in a data warehouse. In The 3rd ACS/IEEE International Conference on Computer Systems and Applications, page 27, 2005.
15. Chuan Zhang, Xin Yao, and Jian Yang. An evolutionary approach to materialized views selection in a data warehouse environment. IEEE TRANS. SYST., MAN, CYBERN, 31:282—294, 2001.
16. Seyed Hamid Talebian and Sameem Abdul Kareem. Using genetic algorithm to select materialized views subject to dual constraints. In 2009 International Conference on Signal Processing Systems, pages 633–638, Singapore, 2009.