Algorithms for data warehouse design to enhance decision-making

ZIYATI ELHOUSSAINE, DRISS ABOUTAJDINE, EL QADI ABDERRAHIM
Mohammed V University GSCM-LRIT
Agdal-Rabat
MAROC
ziyati@gmail.com
aboutaj@fsr.ac.ma
elqadi@est-umi.ac.ma
http://www.fsr.ac.ma/GSCM/

Abstract: - On-line analytical processing (OLAP) queries are strongly affected by the amount data needed to be accessed from the disk. Therefore, there is a need to employ techniques that can facilitate efficient execution of these queries. There has been a lot of work to optimize the performance of relational data warehouses. Among the two fragmentation techniques, vertical fragmentation is often considered more complicated than horizontal, it nearly impossible to obtain an optimal solution. Data partitioning concept that has been studied in the context of relational databases aims to reduce query execution time and facilitate the parallel execution of queries. In this paper, we develop a new framework based on genetic algorithm for applying the partitioning technique on relational DW schema (star schema) to minimize the total query execution cost. We develop an analytical cost model for executing a set of OLAP queries on a partitioned star schema. We conduct experiments to evaluate the utility of partitioning in efficiently executing OLAP queries. Finally, we show how partitioning can be used to facilitate parallel execution of OLAP queries.

Key-Words: - Partitioning, warehouse, OLAP queries, Genetic algorithm, penalty function, query optimization.

1 Introduction

Data warehousing applications typically involve massive and huge data that push database management technology to the limit, and also using complex queries due to the presence of join and aggregate operations. In addition data warehouse integrate massive amounts of data from multiple sources and are primarily used for decision support purposes. They have to process complex analytical queries for different access forms such as OLAP, data mining, etc. Ensuring short query response is enormously difficult and can only be achieved by a combination of different approaches. Several techniques were proposed and supported by commercial systems such as Materialized view, index, sampling and parallel computing technologies each technique cited above represents a research area. Vertical and horizontal [1, 2, 4] partitioning are two non redundant techniques, several work and commercial systems show their utility and impact in optimizing OLAP queries. But they did not give the same interest to the hybrid or mixed partitioning. In this paper we will concentrate on data partitioning aspect and present a new approach of hybrid partitioning based on genetic algorithm with and without penalty [6]. The experiment results using benchmark APB-1 release II benchmark (consisting of the fact table ACTVARS and CUSTLEVEL, CHANLEVEL, TIMELEVEL and PRODLEVEL representing the dimensions tables) show enhancement of the process of partitioning in relational data warehouse environment.

1.1 Contributions and organization of the paper

The proposed work addresses the problem of a relational warehouse fragmentation and proposes a hybrid method (combine horizontal and vertical fragmentations by a genetic algorithm) that minimizes the query processing cost. This paper is divided into six sections: section 2 formulizes the problems in data warehouse modeled using star schema. Section 3 presents the genetic algorithms for the vertical and horizontal fragmentations. Sections 4 and 5 present an implementation of the genetic algorithms for both cases (vertical and horizontal). Section 6 gives the experimental results and concludes the paper by summarizing the results.

2 Vertical and horizontal fragmentation problems

Suppose a relational warehouse modeled by a star schema with d dimension tables and a fact table F. Among these dimension we consider that g tables are fragmented $(g \le d)$. Each dimension table D_i $(1 \le i \le g)$ is partitioned into m_i fragments: $\{D_{i1}, D_{i2}, ..., D_{im_i}\}$, where each horizontal fragment D_{ij} is defined as: $D_{ij} = \sigma_{cl_j^i}(D_i)$ with cl_j^i and $\sigma\,(1 \le i \le g, 1 \le j \le m_i)$ represent a conjunction of simple predicates and the selection operator, respectively, Thus the fragmentation schema of the fact table is defined as follows: $F_i = F \rtimes D_{1i} \rtimes D_{2i} \rtimes ... \rtimes D_{gi}$, $(1 \le i \le m_i)$ with $\rtimes$ represents the semi jointure operation. Using Relational algebra, vertical fragment [11] could be written as $F_i = \pi_{attr(F_i)}(R)$ for all $i \in \{1, ..., k_i\}$ verticals fragmentation replaces R by a set $\{F_1, ..., Fv\}$ of new relations schema such that:

1) The attributes are distributed, i.e $Fu = \bigcup_{i=1}^{v} F_i$,
2) Each relation r_i over F_i is split into relations $r_i = \pi_{F_i}(r)(i = 1, ..., v)$ such that $r = r_i \bowtie ... \bowtie r_v$ holds.
3) In relational algebra, $F_i = F_1 \bowtie ... Fv$

In both cases horizontal and vertical the problem is: Given a set of dimensions tables $D = \{D_1, D_2, ..., D_d\}$ and a set of OLAP queries $Q = \{Q_1, Q_2, ..., Q_m\}$ where each query $Q_i\,(1 \le i \le m)$ has an access frequency. The problem of horizontal (vertical) fragmentation consists in determining set dimensions (attributes) to partition and generate a set of fragments in order to minimize to total queries cost. We first present the genetic algorithm employed in the horizontal and vertical fragmentation and the impact of introducing the penalty function into the process of fragmentation.

3 Genetic algorithms

The literature on genetic algorithms is very rich. There are many variations, but in this section, one present briefly the definition and concepts of a basic genetic algorithm. GA is a search algorithm based on the mechanism of natural selection and natural genetics and is used to search large, non-linear search spaces where expert knowledge is lacking or difficult to encode and where traditional optimization techniques fall short. A GA works with a population of individual strings (chromosomes), each representing a possible solution to a given problem. In this work each position in the chromosome may take on one of a finite set of values, and represents a variable from the user's system. Each chromosome (individual) is assigned a fitness value according to the result of the fitness (or objective) function. Such highly fit chromosomes will survive more frequently than other in the population, and they are given more opportunities to reproduce and the offspring (child) share features taken from their parents. For many problems (manufacturing, communication, neural networks, etc…) genetic algorithms can often find good solutions (near-optimal) in around 100 generations. This can be many times faster than an exhaustive search approaches. GAs judiciously uses the idea of randomness when performing a search. However, it is important to state that genetic algorithm is not a simply random search algorithm. Indeed, random search algorithms can be inherently inefficient due to the directionless nature of their search. GAs are not directionless. They utilize knowledge from previous generations in order to construct a new generation that will approach the optimal solution. In other words, they use past knowledge to direct the search. The main input to a simple GA is a set of all possible points that comprise the search space for solution. Three basic operations that characterize GAs are respectively: selection, crossover and mutation. These operations act on a population in their search for an optimal solution.

GA for vertical fragmentation:

Parameter**:** Population
1: **Begin**

2: Generate the initial population $G(0)$

3: **Repeat**

4: $t = t + 1$

5: $G_1(t) \leftarrow UniformCrossover(G(t-1))$

6: $G_2(t) \leftarrow Mutation(G_1(t))$

7: $G(t) \leftarrow G_2(t)$

8: **Until (**termination condition is satisfied)

9: **End**

3.1 Selection

Selection is a process in which chromosomes are copied according to their fitness function value. There are many methods for selecting the best chromosome, the roulette wheel method is used in both algorithms: each chromosome is associated with its fitness value calculated using the cost model defined later, so the chromosome with high fitness values has chances to be selected.

3.2 Crossover Operation

The traditional crossover operator randomly by selecting genes from parent chromosomes. In both situations we selected a two-point crossover mechanism to gives the same chances to the attributes with high and low number of sub domains (e.g. Season that has 4 sub domains) will have a probability greater than gender (two sub domains).

Crossover algorithm:

Input: Generation G

Parameter: crossover probability P_c

1: **Begin**

2: Select pair chromosomes of G randomly (I_i, I_j)

3: Sample $n \in [1, d]$

// choose the n^{th} dimension;
// in both individuals

4: Sample $u \in [0,1[$;

5: If $(u < P_c)$ then

6: $swap(I_i(m), I_j(m))$

// change partition
// value from m^{th} value

7: End if

8: **End**

For example:

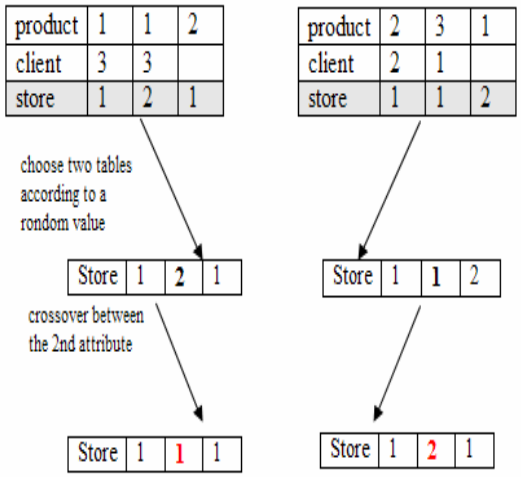

Fig.1 crossover operation during vertical partitioning

3.3 Mutation

After a crossover is performed, the resulting solution might fall into a local optimum. Mutation is needed to create new genes (according to some user-defined probability) that may not be present in any number of a population and enable the algorithm to reach all possible solutions in the search space. When creating a new generation by crossover and mutation the best chromosome might be lost. Elitism is a method which first copies the best chromosome (s) to the new population (from $G(t)$ *to* $G(t+1)$) and this before crossover and mutation application. Elitism rapidly increases the performance of the GA, by preventing loss of the best-found solution. Example for the horizontal fragmentation:

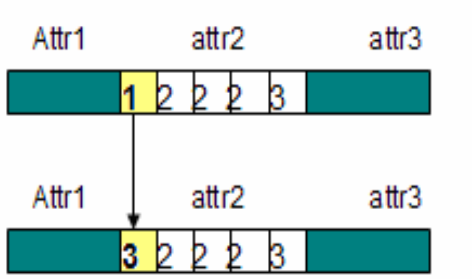

Fig.2

Initial chromosome has 3 fragments concerning attr2. After mutation process the resulting chromosome has only 2 fragments, almost the same for the vertical partitioning, mutation algorithm is:

Input: Generation

Parameter: mutation probability P_m

1: **Begin**

2: choose an individual from G randomly

3: repeat

4: sample $(m \in]0,1[)$

4: If $(m < P_m)$

5: mutate the # fragment

6: end if

7: until **number_of_mutation**

8: **End**

For example:

product	1	3	2
client	3	3	
store	1	2	1

product	1	3	2

product	1	2	2

Fig.3

We have in the first case 3 vertical fragments, but, in the second chromosome we have two (attr1 is located in site A and attr2 and attr3 in site B). In the same way, mutations could occur on several attributes of the individual.

3.4 Genetic algorithm for the mixed fragmentation

In this section, we discuss an algorithm for fragmenting a star schema with one fact table F and d dimension tables $\{D_1, D_2, ..., D_d\}$. This algorithm partitions dimension tables first and then use their fragmentation schemas to derive the fragments of the fact table. To partition a dimension table, we use the selectivity factors, the frequency of each query accessing this table and selection predicates defined on this dimension table. A simple predicate p is defined by:

$$p : A_i \ \theta \ value$$

Where A_i is an attribute, $\theta \in \{=, <, \leq, >, \geq, \neq\}$, $value \in Dom(A_i)$.

The input is to the proposed algorithm are a set of dimension tables d and one fact table, and a set of most frequently asked OLAP queries $Q = \{Q_1, Q_2, ..., Q_m\}$ with their frequencies. The main steps of the algorithm are:

- Enumerate all simple predicates used by OLAP Queries $\{Q_1, Q_2, ..., Q_m\}$
- Assign to each dimension table $D_i (1 \leq i \leq d)$ a set of its simple predicates SSP^{D_i}.
- Each dimension table having $SSP^{D_i} = \varnothing$ can not fragmented.
- Application of COM_MIN algorithm [16,4] to the simple predicates to $D_{candidates}$ (where $SSP^{D_i} \neq \varnothing$)

Starting by the horizontal fragmentation:

For example, Consider two fragmentation attributes Age and Gender of dimension table customer, the domain of these attributes are defined as:

$Dom(age) =]0,120[$, $Dom(Gender) = \{'M','F'\}$

The domain of this attribute $(]0,120[)$ is then partitioned into three sub domains: $d_{11} =]0,18]$, $d_{12} =]18,60[$ and $d_{13} = [60,120]$.

Similarly the domain of Gender attribute is decomposed into two sub domains: $Dom(Gender) = d_{21} \cup d_{22}$ .Each fragmentation attributes A_i can be presented by an array with n_i cells, where n_i corresponds to number of its sub domains. The values of these cells are between 1 and n_i . If two cells have the same values, then they will be merged to form only one. The Horizontal fragmentation will be presented by multidimensional array (each array present a fragmentation attributes) [7].
For example we define two fragmentations attributes with there domains figure 1:

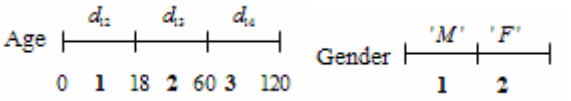

Fig.4

An example of a possible solution is:

Age	**1**	**2**	**1**
Gender	**1**	**1**	

Table 1

In table 1 we can deduce that the fragmentation of the data warehouse is not performed using the attribute **gender** because all its sub domains have the same value. consequently; the warehouse will be fragmented using age attribute,
$P1: age < '18' \vee age > 60$ and $P2: 18 \le age \le 60$.
Note that if we define another fragmentation attribute each fragment is represented by a conjunctive of simple predicates. Also the derived horizontal fragmentation is defined as follow: given a two relation R and S , with S containing foreign key of R let R be horizontally partitioned into set of $HFs\{R_1, R_2, ..., R_m\}$, and then S can derived horizontally partitioned into $HFs\{S_1, S_2, ..., S_m\}$, where each S_i is given by : $S_i = S \triangleright\!\!< R_i$ where $(1 \le i \le m)$, and $\triangleright\!\!<$ is semi jointure [9]. The proposed coding satisfies the correctness rules (completeness, reconstruction and disjointness): the completeness ensures that all tuples of a relation are mapped into at least on fragment without any loss. The completeness of the dimension tables is guaranteed by the use of COM_MIN algorithm [16, 8]. The completeness of the derived horizontal fragmentation of the fact table is guaranteed as long as the referential integrity among the dimension tables and the fact table. The reconstruction ensures that the fragmented relation can be reconstructible from its fragments [16]. In our case, the reconstruction of the fact and the dimension tables are obtained by union operation $F = \bigcup_{i=1}^{N} F_i$ (Horizontal fragmentation).The disjointness ensures that the fragments of a relation are non-overlapping. For the fragments of the fact table, the disjointness rule is guaranteed by the fact that any HF of the fact table has to be joined with only one HF of a dimension.
Then for the vertical fragmentation, we should:

1. Extract attributes from the queries (select clause).
2. Evaluate the frequency access for each query.

The most important part to apply GA is chromosome. As known coding the chromosome is crucial. To code the VP schema, we adapt the coding used in HF. Let $D_i = \{A_1, A_2, ..., A_{m_i}\}$ be a dimension table with m_i attributes, where A_1 is the primary key. Each fragmentation scheme can be presented by an array those cells are between 1 and m_i . The first cell represents the first attribute. If two or several cells have the same values, this means that the attributes from one vertical fragment. To illustrate this coding:
The chromosome is presented in table 2 as above (multi-dimensional arrays) for example:

Product	**1**	**2**	**1**
Client	**1**	**1**	
Store	**1**	**1**	**0**

Table 2

In table1 (product dimension), *attribute1* and *attribute3* will be in site A and *attribute2* in the site B and so on for other dimensions (Client, Store). We present a general analytical cost model for processing a query over unpartitioned and partitioned warehouse is given [4] as:

$$Cost(Q_k) = \sum_{j=1}^{N} valid(Q_k, S_j) \prod_{i=1}^{M_j} \left[\frac{(Sel_F^{p_i} \times \|F\| \times L)}{PS} \right] (1)$$

where, M_j , F , L and PS represent respectively the numbers of selection predicates defining the

fact fragment of the sub star schema S_j, the cardinality of the fact table(number of tuples) F,the width, in bytes, of a tuple of a table F and the page size of the file system, respectively. And also

$$valid(Q_k, S_j) = \begin{cases} 1 \text{ if the sub star schema } S_i \text{ is needed for } Q_k \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

The total cost of executing a set of queries Q is given by: $TC(Q) = \sum_{j=1}^{m} Cost(Q_j)$ (3).

The selectivity factors are chosen using an uniform distribution UD.

3.4.1 Penalty function

During this process so many infeasible solutions may be found, but ignored, simply because they do not respect the constraint (e.g. their cost is too large or space constraint) that why, we should introduce a penalty function to the fitness function. There are 3 forms:

Subtract Mode (S)

- $f'(x) = f(x) - Pen(x)$ *if* $f(x) \geq Pen(x)$
- $f'(x) = 0$ *otherwise*

Divide Mode (D)

- $f'(x) = \dfrac{f(x)}{Pen(x)}$ *if* $Pen(x) > 1$
- $f'(x) = f(x)$ *if* $Pen(x) \leq 1$

Subtract and divide mode

$$f'(x) = f(x) - Pen(x) \text{ if } f(x) > Pen(x)$$

$$f'(x) = \frac{f(x)}{Pen(x)} \text{ if } Pen(x) > 1 \text{ and } f(x) \leq Pen(x)$$

$$f'(x) = f(x) \text{ if } Pen(x) \leq 1 \text{ and } f(x) \leq Pen(x)$$

Where

$f'(x)$: the new objective function

$f(x)$: the old objective function

$Pen(x)$: the penalty function

There penalty functions also have 3 forms:

Logarithmic penalty (LG):

$$Pen(x) = \log_2(1 + \rho(seuil - S))$$

Linear penalty (LN):

$$Pen(x) = 1 + \rho(seuil - S)$$

Exponential penalty (EX):

$$Pen(x) = (1 + \rho(seuil - S))^2$$

4 Problem solution

The proposed method for solving this problem of fragmentation in a relational data warehouse to minimize the execution time of OLAP queries is a mixed or combined fragmentation based on adaptative Genetic Algorithm (in both cases vertical and horizontal fragmentation):
The algorithm is presented as:

Input:

Set of Queries: Q

Set of a data warehouse dimensions: D

Output: vertical and fragmentation schemes.

Begin

Extract (access frequencies, simple predicates)

Start_*horizontal*_genetic_fragmentation ()

Start_*vertical*_genetic_fragmentation ()

// fragment the warehouse horizontally

// using the generated schemes

// in both cases

end

Figure below describe different step during this process

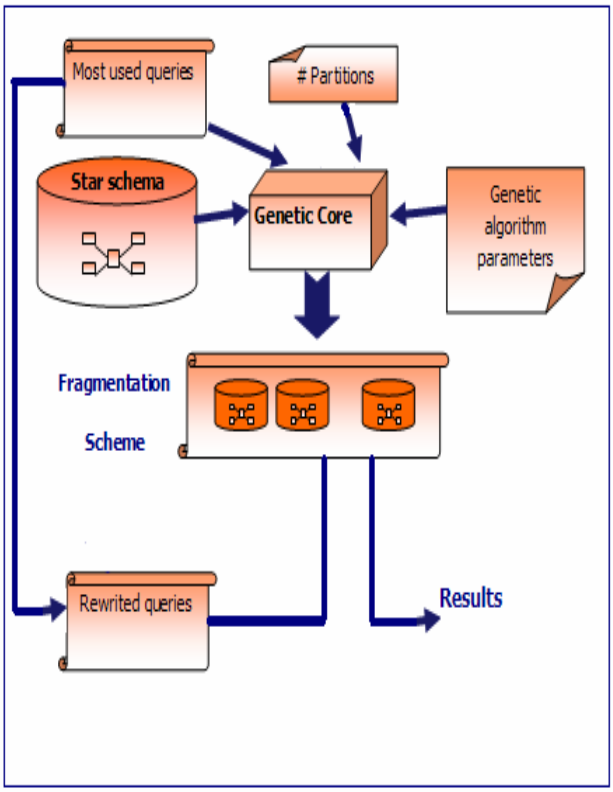

Fig.5 proposed Genetic algorithm schema

5 Experimental performance analysis

In order to analyze and test the performance and convergence of the genetic algorithms, we use the dataset from the APB-1 benchmark [3], The star schema of this benchmark has one fact table Actvars and four dimensions:

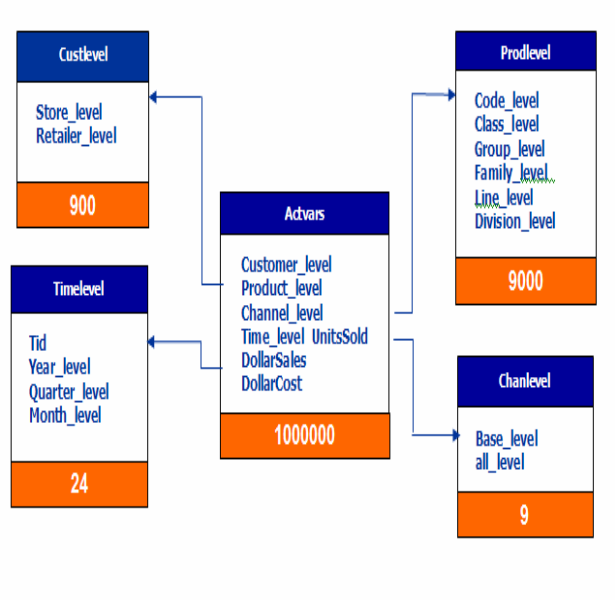

Fig.6 data warehouse schema

Table	Number of tuples	Width of a tuple
Actvars	1 000000	74
Chanlevel	9	24
Custlevel	900	24
Prodlevel	9000	72
Timelevel	24	36

Table 3 sizes of tables

This warehouse has been populated using the generation module of APB1. This warehouse has been installed under oracle 10g on a Pentium *IV* 1.8 GHz (with a memory of 256 Mo, 60Go) running under windows XP pro. The vertical fragmentation, programs developed in java, all our experiments run on Pentium. We have considered a set of OLAP queries [12]. Each query has selection predicates, where each one has its selectivity, for example:

Query 1:

```
SELECT
        prodlevel.group_level,
        custlevel.retailer_level,
        chanlevel.base_level,
        timelevel.month_level,
        sum(a.unitsolds), sum(a.dollarsales)
FROM
        dw.actvars a, dw.prodlevel, dw.timelevel, dw.custlevel, dw.chanlevel
WHERE
          a.product_level = prodlevel.code_level
        and a.customer_level = custlevel.store_level
        and a.channel_level = chanlevel.base_level
        and a.time_level = timelevel.tid
        and prodlevel.line_level = 'C8IO245PU0W5'       selectivity = 1/15
        and custlevel.retailer_level = 'AZC10JUN5NO0'   selectivity = 1/99
        and chanlevel.all_level = 'BCDEFGHIJKLM'        selectivity = 1/9
        and timelevel.month_level between '7' and '9'   selectivity = 1/12
GROUP BY
        prodlevel.line_level,
        custlevel.retailer_level,
        chanlevel.all_level,
        timelevel.month_level
```

Query 2:

```
SELECT
        base_level,
        sum(dollarsales) ,
        sum(UnitsSold)
FROM
        dw.actvars,
        dw.timelevel,
        dw.prodlevel,
        dw.chanlevel
WHERE
        time_level = tid
        AND product_level = code_level
        AND channel_level = base_level
        AND month_level between '10' and '12'          selectivity = 1/12
        AND family_level = 'AHKBHFENPPJ8'              selectivity = 1/75
        AND all_level = 'CDEFGHIJKLMN'                 selectivity = 1/9
GROUP BY
        base_level;
```

Query 3:

```
SELECT
        base_level,
        sum(dollarsales) ,
        sum(UnitsSold)
FROM
        dw.actvars,
        dw.timelevel,
        dw.prodlevel,
        dw.chanlevel
WHERE time_level = tid
        AND product_level = code_level
        AND channel_level = base_level
        AND month_level between '10' and '12'   selectivity=1/12
        AND family_level = 'AHKBHFENPPJ8'       selectivity=1/75
        AND all_level = 'CDEFGHIJKLMN'          selectivity=1/9
GROUP BY
        base_level
```

Query 4:

```
SELECT
        prodlevel.code_level,
        prodlevel.family_level,
        chanlevel.all_level,
        sum(actvars.dollarsales),
        count(actvars.dollarsales),
        count(*)
FROM
        dw.actvars,
        dw.chanlevel,
        dw.prodlevel
WHERE
        chanlevel.base_level = actvars.channel_level
        and prodlevel.code_level = actvars.product_level
        and (prodlevel.family_level = ' AOS616MICD5U')   selectivity = 1/75
        and (chanlevel.all_level = ' BCDEFGHIJKLM')      selectivity = 1/9
GROUP BY
        prodlevel.code_level,
        prodlevel.family_level,
        chanlevel.all_level
```

Query 5:

```
SELECT
        prodlevel.family_level,
        timelevel.month_level,
        chanlevel.base_level,
        chanlevel.all_level,
        sum(actvars.dollarsales),
        count(actvars.dollarsales),
        sum(actvars.unitssold),
        count(actvars.unitssold),
        count(*)
FROM
        dw.actvars,
        dw.chanlevel,
        dw.timelevel,
        dw.prodlevel
WHERE
        chanlevel.base_level = actvars.channel_level
        and timelevel.tid = actvars.time_level
        and prodlevel.code_level = actvars.product_level
        and (prodlevel.family_level = ' A9Q2JYTPY6PF')   selectivity=1/75
        and (timelevel.month_level = '01')               selectivity=1/12
        and (chanlevel.all_level = ' ABCDEFGHIJKL')      selectivity=1/9
GROUP BY
        prodlevel.family_level,
        timelevel.month_level,
        chanlevel.base_level,
        chanlevel.all_level;
```

In the first we compare the case where using only the horizontal fragmentation and no the case where no fragmentation figure 7, this result is proved by [4].

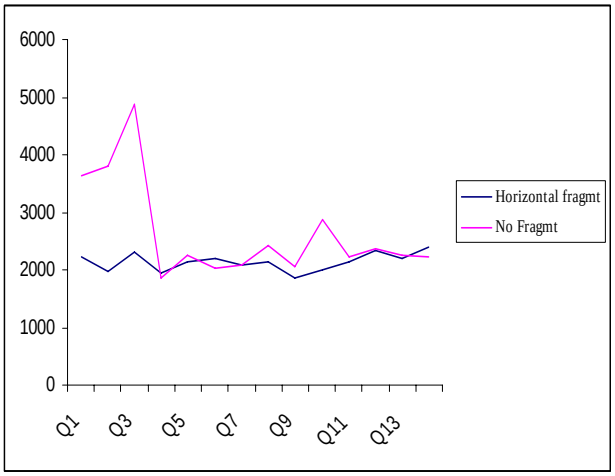

Fig.7

And in figure 8 we study the mixed fragmentation and its usefulness against horizontal fragmentation. We can see that it present good solution more than using only HF.

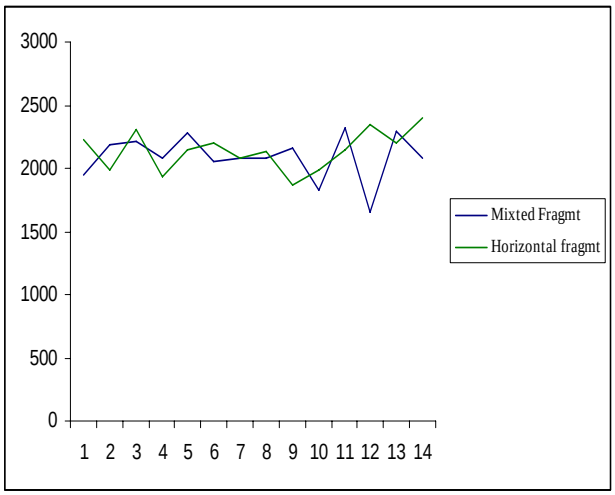

Fig.8

Also we want to see the impact of the numbers of vertical fragments to the performance figure 9.

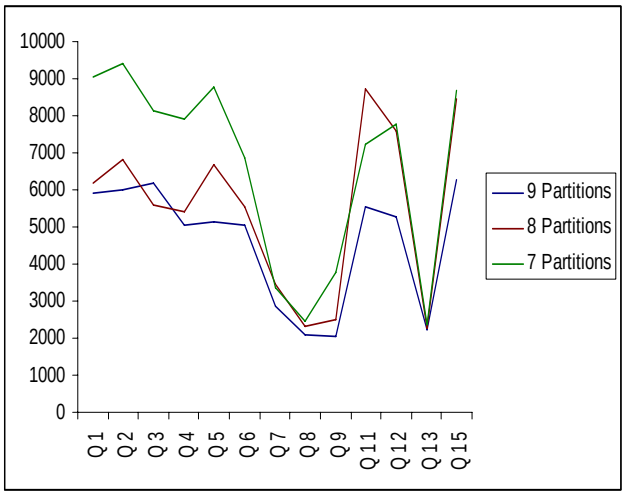

Fig.9

And utility of introducing a penalty function [5, 6] to some infeasible solutions during the algorithm figure 10.

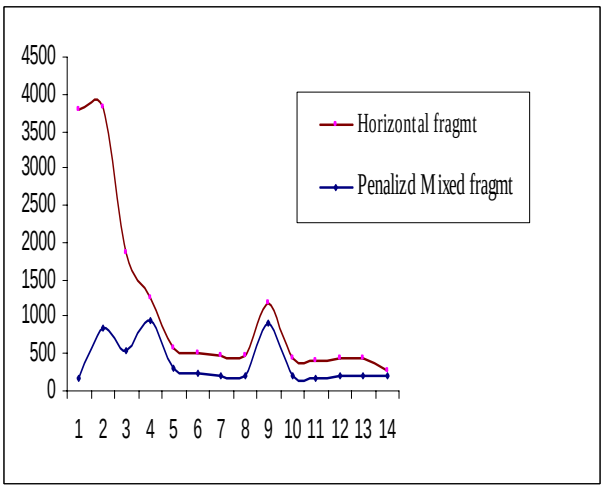

Fig.10

6 Conclusions

As a network service, a data warehouse system collects data from different remote data sources and disseminates high-quality data analysis to decision makers locally and remotely, also data warehouse (DWs) become larger and larger make the amount of the time taken to execute to complex OLAP gets larger. Hence, there is a need for developing techniques that can facilitate efficient OLAP query execution for large DWs. In this regard, we take a fresh look at data partitioning and show its utility in efficiently executing OLAP queries. In this paper we present a complete algorithm for vertical and horizontal fragmentation using genetic algorithm we showed that plays a significant role in design of a data warehouse system using our cost model for evaluating the cost of frequently queries performed in a top of the partitioned relational data warehouse scheme in order to enhance physical design. Our experiments results show that our method can provide a significantly better solution than previous algorithms in terms of minimization of query processing cost [4, 7]. There are many other future works especially query nature (use aggregate function SUM, AVG, COUNT, STDDEV, VAR…..).

References:

[1] A. Y. Noaman and K. Barker. A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse. *In the 8th International Conference on Information and knowledge Management (CIKM'90)*, November 1999, pages 154-161.

[2] A. Sanjay, V. R. Narasayya, and B. Yang. Integrating vertical and horizontal partitioning into automated physical database design. Proceedings of the ACM SIGMOD International conference on Management of Data, June 2004, pages 359-370.

[3] OLAP Council. Apb-1 olap benchmark, release ii. *http://www.olapcouncil.org/research/bmarkly.html* 1998.

[4] L. Bellatreche, K. Boukhalfa, and H. I. abdalla. Saga: A combination of combination of genetic and simulated annealing algorithms for physical data warehouse design. *In 23rd British National Conference on databases*, July 2006, (212-219).

[5] C. Zhang et J. Yang. Genetic algorithm for materialized view selection in data warehouse environments. Proceeding of the International conference on Data Warehousing and knowledge discovery (DAWAK'99), September 1999, pages 116-125.

[6] Yu, J. X., C.-H. Choi, et G. Gou (2004), Materialized view selection as constrained evolution optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, Part 3 33(4), pp 448-467.

[7] L. Bellatreche, K. Boukhalfa, La fragmentation dans les entrepôts de données une approche basée sur les algorithmes génétiques, *Revue des nouvelles Technologies de l'information (EDA'2005)*, juin 2005, pages 141-160.

[8] S. Ceri, M. Negri, and G. Pelagatti. Horizontal data partitioning in database design. Proceedings of the ACM SIGMOD International Conference on Management of Data. SIGPLAN Notices, pages 128–136, 1982.

[9] Z. ELhoussaine, K.boukhalfa, L.Bellatreche Sélection de structures d'optimisation non redondantes dans les entrepôts de données relation. The 8th International Symposium on programming and Systems (ISPS'2007)

[10] A. Paterson. The design and development of social science data warehouse: A case study of the human resources development data warehouse project of the human sciences research council, South Africa. Data science journal, 2:12-24, February 2003.

[11] M. Gorlfarelli, S. Rizzi. Designing the data warehouse: key steps and crucial issues, *journal of computer science and Information Management*, vol. 2, n. 3, 1999.

[12] A. Gupta, V. Harinarayan, and D. Quass. Aggregate query processing in data warehousing environments, Proc. 21st Very Large Database conf. (VLDB95), 1995, pp. 358-369.

[13] M. Mitchell, S. Forrest and J. Holland. The Royal Road Algorithms: Fitness Landscapes and GA Performance. Toward a practice of autonomous Systems : Proceedings of the first European Conference on Artificial Life, Cambridge, MA, MIT Press, 1991.

[14] J.Windom. Research problems in data warehousing Proceeding of the fourth International Conference on Information and Knowledge Management (CIKM), Baltimore, MD, November 1995,pp. 25-30.

[15] J. Muthuraj, S. Chakravarthy, R. Varadarajan, and S. B. Navathe, A formal approach to the vertical partitioning problem in distributed database design, in Proc. The second International Conference on Parallel and Distributed Information Systems, San Diego, CA, USA, Jan 1993.

[16] M. T. Özsu and P. Valduriez. *Principals of Distributed database.* Prentice Hall, 1991.

[17] A. Datta, B. Moon, and H. Thomas. A case for parallelism in data warehousing and OLAP. In the 9th international workshop on database and expert system applications (DEXA98), pages 226-231, August 1998.

[18] L. Bellatreche, M. Schneider, H. Lorinquer, and M. Mohania, "Bringing together partitioning, materialized views and indexes to optimize performance of relational data warehouses," Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2004), pp. 15–25, September 2004.

[19] S. Papadomanolakis and A. Ailamaki, "Autopart: Automating schema design for large scientific databases using data partitioning," Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), pp. 383–392, June 2004.