A data mining-based approach for data warehouse optimisation

AMIRAT Hanane
Computer Sciences Department
University of Laghouat
Laghouat, ALGERIA
h.amirat@mail.lagh-univ.dz

BOUKHALFA Kamel
Computer Sciences Department
USTHB
Algiers, ALGERIA
kboukhalfa@usthb.dz

***Abstract*—** Data warehouses are currently form a good basis of decision support systems. The main characteristics of these data warehouses are their large size and complexity of its decision-support queries. Many optimization techniques have been proposed to reduce the execution cost of these queries (indexes, materialized views, partitioning, etc). Several research works have been proposed in the literature to handle the selection problems of these techniques during the physical design phase, using heuristics: meta-heuristics, linear programming, data mining techniques, etc. We focus in this paper on combined selection of horizontal partitioning and bitmap join indexes. All the proposed approaches use algorithms to share attributes between these two techniques. We show in this paper that the approaches based on attributes sharing can ignore some interesting solutions. We propose, in this paper, a new approach based on data mining techniques. It consists of sharing queries between horizontal partitioning and bitmap join indexes. Each subset of queries will be exploited by the suitable optimization technique to select the appropriate optimization configuration. The queries sharing allows the pruning of the search space and the reduction of the complexity of selection problems. To validate our approach, we conducted an experimental study on a real data warehouse under the Oracle 11g DBMS. We have also compared our approach with a state-of-the-art work.

Keywords- Data warehouses, Physical design, Horizontal partitioning, Bitmap join indexes, Data mining.

I. INTRODUCTION

A data warehouse stores large amounts of consolidated and historical data. It is specially designed to answer complex decision-support queries [12]. Those queries involve several complex join and aggregates operations at the same time which induce an expensive execution cost. To improve the performance of these queries, the data warehouse administrator has to select during the physical design phase one or multiple optimization techniques. Several techniques have been proposed such as: materialized views [24], index [10], horizontal Partitioning [18], etc. Horizontal Partitioning (HP) and Bitmap Join Index(BJI) are two widely used techniques in recent years for data warehouse optimization. In [3], authors show that HP and BJI are two similar techniques since they optimize star join operations and share the same resource: the selection attributes which are the non-key attributes of dimension tables. However, the isolated selection of HP and BJI doesn't allow exploitation of the similarities between these two techniques. Indeed, the selection of each technique is NP-hard problem [4] [9] and their combination increases the complexity of the problem. In the literature, few works are dealing with the combined selection of HP and BJI have been proposed [20] [7] [6] [16]. All these approaches are based on sharing selection attributes between HP and BJI. Our idea is to focus on the queries workload since we aim to optimize it. We propose to share queries between HP and BJI, according to specific criterion. We aim by this sharing to reduce the size of the input query workload for each selection process, and thereby it will be possible to significantly prune the selection problem.
The authors in [29] have assert that the execution time of physical design tools such as Microsoft Tuning Wizard and DB2 Index Advisor increases exponentially with linear increase in the size of the query workload. Therefore, the number of queries has an impact in the physical design phase [8]. However, data mining techniques are being used currently to reduce the complexity of selection problems and improve scalability, therefore we propose in this paper to use data mining techniques in HP and BJI selection problem. The paper is organized into six sections. We formalize in Section 2 the HP and BJI selection problem. We present in Section 3 existing related work to solve this problem. In Section 4 we present our new approach and then we experimentally study its efficiency in Section 5. We conclude the paper in Section 6.

II. HP AND BJI SELECTION PROBLEM FORMALIZATION

The horizontal partitioning and bitmap join indexes selection problem can be formulated as follow:
For a given data warehouse with a set of d dimension tables $D=\{D_1,D_2,...,D_d\}$ contain qualitative data, a fact table F contains foreign keys of the dimension tables in addition to a set of collected measures, and a query workload $Q = \{Q_1,Q_2,...,Q_m\}$. Let's consider the thresholds S and W where S is the storage space quota for indexes and W represents the maximum number of fact fragments.
HP and BJI selection problem consists in selecting a HP schema (PS) and BJI configuration (*IC*) that optimize the execution cost of the workload *Q* with $Size(IC) \leq S$ and $N(PS) \leq$

W where *Size(IC)* and *N* represent the disk space occupied by *IC* and the number of fact fragments respectively.
Based on this formalization, we note that this problem is a combination of two difficult sub problems: index configuration and partitioning schema selection (which are known as NP-hard problems [4] [9]).

III. RELATED WORK

Most existing works propose isolated selection of HP [4] [2] [21] [11] or BJI [1] [5] [28] [14] [26] [27]. Select one technique is generally insufficient since some categories of queries are not optimized with this selection. Therefore, few research studies are dealing with simultaneous index and horizontal partitioning selection [20] [7] [6] [16].
Stohr et al. [20] propose a combination of horizontal partitioning with bitmap join index and parallel processing for the design of parallel data warehouses. Their approach requires only one attribute per dimension table to be used in the partitioning of the fact table. The BJI selection is done on HP unused attributes. This fact results a large number of candidate indexes.
In [7], the authors propose to use HP to prune the search space of BJI selection problem. Their proposal consists of starting by partitioning the data warehouse, and then selecting the BJI set to optimize the queries that do not get benefit from HP (called non benefit queries).
Bouchakri et al. [6] propose to use the clustering algorithm *k-means* to share all selection attributes extracted from the query workload between HP and BJI and then select each optimization technique with a genetic algorithm.
Finally, the authors in [16] have proposed a combined selection of HP and BJI based on multi-agent system. Their approach is composed of set of agents. One of those agents is responsible to share the selection attributes between the optimization techniques based on the work of [6]. The selection of each technique is provided by a set of agents using a genetic algorithm.

Existing studies related to HP and BJI are consisting to assign each candidate selection attributes to a single optimization technique at once. However, this strategy may leads, in some cases to insufficient optimization. To illustrate this, we suppose a data warehouse start schema composed of fact table Sales, and three dimension tables: *Product*, *Customer* and *Time*. Let's assume that we partition the table *Customer* into two partitions according to the attribute *City* as follow:

- $Customer1=\sigma_{(City=Ouargla)}(Customer)$
- $Customer2=\sigma_{(City\neq Ouargla)}(Customer)$

Then we partition the table *Sales* into two partitions (referential partitioning):

- Sales1=Sales ⋉ Customer1
- Sales2=Sales ⋉ Customer2

Where ⋉ represents the semi-join operator. Let's suppose that the partition *Sales1* contains 10% of Sales tuples and *Sales2* contains the rest (90%). Let's suppose a query containing the predicate *City='Oran'*, where 'Oran' is another city and 5% of sales that have been sold to customers living in *'Oran'*. To execute this query, all tuples of the partition *Sales2* must be loaded in memory (90% of Sales tuples), which will generate a high cost. To reduce it, it will be interesting if we select a BJI index on the attribute *City*. Using this index, only tuples referencing the city 'Oran' are loaded (5%). In all related work cited above, if the attribute *City* is used to fragment the data warehouse, it cannot be used in BJI selection process. These approaches concentrate on the selection attributes to optimize the queries workload. We believe that it will be important to focus on queries and not attributes by sharing them between HP and BJI in order to define the best technique to use for a given query according to some factors. In addition, our approach allows the use of the same attribute to fragment and index the data warehouse at the same time in order to improve the performance, which is impossible in existing works.

IV. OUR APPROACH

We propose, in this paper, a new approach for the combined selection of HP and BJI using data mining techniques. The general architecture of our selection process is shown in Figure 1.

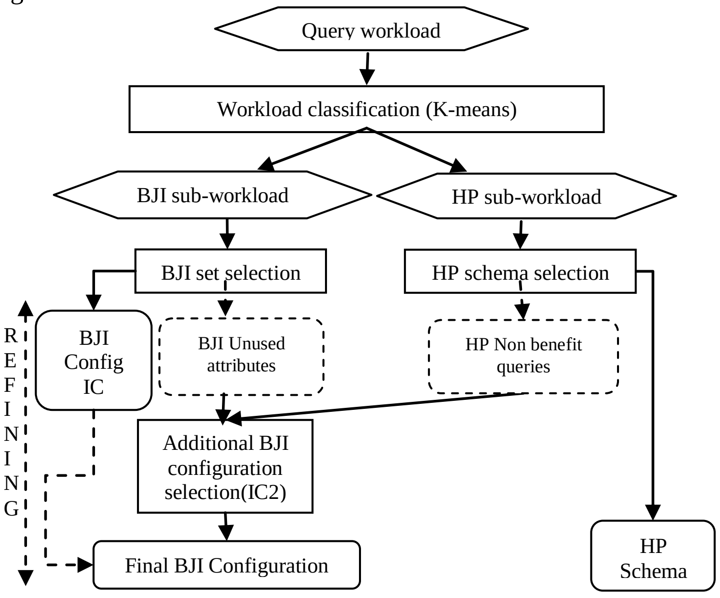

Figure1: Our approach architecture

Our approach is based on the classification of queries. It consists of sharing the global query workload *Q,* with unsupervised classification algorithm *k-means,* into two sub-workloads named: *HP_Workload* and *BJI_Workload*. The classification weight is calculated from two criterions: the cardinality of selection attributes and the query selectivity. We'll present in the following sections the detail of our approach.

A. Query workload classification

In order to share the query workload between optimization techniques HP and BJI, we have used the algorithm *k-means* for unsupervised classification (clustering). This algorithm uses, a distance such as the *Euclidean distance* as an intra-class distance. Depending on the number of iterations, *k-means* recalculates the two centers and reassigns every query to the suitable class according to the distance. To adapt this algorithm to our need, we have considered the following:

- The data to classify are the queries of the query workload *Q*.
- Parameter *k* of *k-means* is fixed to 2.
- Queries are classified into two classes: *BJI_Workload* and *HP_Workload*.
- The queries are represented in two-dimensional space. There coordinates are based on a classification weight to determine for each query, if it will be optimized by HP or BJI.

The weight of a given query Q_i is based on two important factors: the cardinality of selection attributes (CAR) and query selectivity (SEL). The classification weight is given by formula below:

$$\text{Weight}(Q_i)= CAR(Q_i) + SEL(Q_i)$$

Attributes Cardinality CAR: It represents is the sum of the cardinalities of the selection attributes used in Q_i.

$$CAR(Qi) = \sum_{i=1}^{m} ||A_i||$$

Where ||Ai|| represents the cardinality of the attribute A_i and *m* is the number of selection attributes in Q_i. The cardinality of an attribute is a very important factor for the selection of a BJI. Depending on it, indexing may be more or less efficient. If the cardinality is very large, an index degenerates toward a sequential scan (of the index structure itself) [22]. For this, we considered that if the sum of a query attributes cardinalities is small, the BJI are more appropriate to optimize this query. Otherwise, the HP is better.

Query Selectivity (SEL) : It represents the fraction of selected table tuples during the query execution. In this paper, we focus on selectivity of queries on the fact table. Let us consider the query Q_1 introduced a simple selection predicate as follows:

Q_1: Select Count (*) From Sales S, Product P Where S.Pid = P.Id And P.Brand ='BMW'.

The selectivity of Q_1 is related to the predicate P_1(*P.Brand='BMW'*). It is equal to the number of tuples of the fact table *Sales* that involve the predicate P_1. The general form of the calculation of a given predicate selectivity P_i sets to a dimension table D_j and required a join operation between D_j and the fact table is given by:

$$\text{Sel}(P_i)= \frac{\text{Selected tuples}}{||F||}$$

Where: ||F|| represents the number of tuples of the fact table F. Let's suppose that the number of selected rows is 71600 tuples and the size of the fact table Sales is ||Sales|| = 6 millions. The selectivity of Q_1 is equal to:

$$SEL(Q_1)=Sel(P_1)=\frac{Selected\ tuples}{||Sales||} = \frac{71600}{||Sales||} = 0.02$$

In the example above, we have calculated the query selectivity if it contains a single selection predicate of the form *Ai = value*. In the case where Q_i contains several selection predicates (which is the case for star join queries), the selectivity is depending on the operators connecting these predicates. The majority of RDBMS consider that if the conditions in a query are independent, the selectivity of each condition can be calculated separately. This logic works globally if the attributes are independent but when they are correlated, the results are wrong. Note that this problem can be avoided with the use of dynamic sampling[13].

The indexes provide a best gain in queries with high selectivity because they select a small number of tuples, and therefore access to a high number of tuples is avoided. We conclude that queries with high selectivity encourage indexing. However, queries with a low selectivity encourage partitioning.

We have noticed that the values for each factor had a different scale. For the criteria CAR, which represents the attributes cardinality, let's suppose that the cardinality of the attribute *City* is 60. However, the query selectivity is expressed as a value between 0 and 1. The direct sum of these two factors shows that CAR is the dominant factor. To get the classification weight consistent, we have performed a normalization of values. We propose to transform each factor values so they follow the standard normal distribution with average 1 and standard deviation 0. Let's consider G to be a sample composed of $g1, g2,...,gn$. Each element of G is calculated as follow:

$$g = \frac{gi - m}{e}$$

Where: *m* is the average and *e* is the standard deviation formulated as:

$$m = \frac{\sum_{i=1}^{n} gi}{n}$$

$$e=\sqrt{(\frac{1}{n}\sum_{i=1}^{n} gi^2) - m^2}$$

For example, let's consider a set of query {Q1,.....,Q10} chosen randomly form our query workload, the calculation of their weight is presented in table 1.

Query	Selectivity (SEL)	Cardinalities sum(CAR)	Normalized SEL	Normalized CAR	Weight
Q1	0.058	12	-1.75	-1.94	-3.69
Q2	0.29	4	4.65	-2.067	2.588
Q3	0.22	6	2.739	-2.03	0.70
Q4	2.98E-5	182	-3.36	0.697	-2.67
Q5	3.68E-5	167	-3.36	7.45	4.085
Q6	0.23	4	2.97	-2.06	0.909
Q7	9.039E-5	121	3.365	-0.25	-3.61
Q8	0.176	4	1.48	-2.067	-0.58
Q9	0.245	4	3.367	-2.067	1.299
Q10	4.172E-7	4,17	-3.368	4.346	0.978

Table1: Queries weight calculation

After applying the algorithm *k-means*, we have obtained two sub-worload: HP_Workload {Q2,Q3,Q5,Q6,Q8,Q9 ,Q10} and BJI_Workload {Q1,Q4,Q7}.

B. *Partitioning schema selection*

To select the partitioning schema, we have used the approach proposed in [4]. It consists in extracting the selection predicates and then decompose the domain of each selection attribute into *K* sub-domains. The sub-domain Sd_i where $i=\{1...K-1\}$ corresponding to the attribute values used by the workload (*HP_Workload* in our case). The last sub-domain Sd_k corresponds to all remaining values. In addition, this strategy is based on a particular coding of partitioning schema where each partitioning schema is represented by a multidimensional array and each row represents an attribute and its sub domains. The value of each cell of a given array representing an attribute is within ($1...n_i$), where n_i represents the number of sub-domains of the attribute A_i. The cells with the same value will be grouped in the same partition. If all the sub domains of an attribute have the same value, the attribute will not participate in the partitioning process. Finally, a genetic algorithm (GA) is used to select a near-optimal partitioning schema. Each chromosome has composite genes and each composite gene contains *n* simple genes. A composite gene in our case represents all sub-domains of a selection attribute extracted from *HP_Workload*, whereas a gene is a sub-domain.

The GA apply three genetic operations: selection, crossover and mutation, to transform the initial population of chromosomes, with the objective to improve their quality.

The evaluation of a problem solution represented by a chromosome is performed with an objective function (*fiteness*) based on a cost model. This cost model is mainly based on the advanced model proposed by the authors in [4]. This model is used to calculate the number of Input/output needed to run a query.

```
Algorithm 1:Genetic algorithm
Begin
1.  Random generation of a population contains m chromosomes X
2.  Evaluation of each chromosome's fitness f(X)
3.  Create new population
      - Select 2 parents chromosomes
      - Crossovers the 2 parents with specific probability $T_c$ to obtain 2 children.
      - Select and mutate a chromosome with specific probability $T_m$.
      - Add new chromosomes to the population
4.  Compose the new population
5.  If the population isn't sufficient repeat from step 2
End
```

C. *Index set selection*

The BJI selection method we propose (see Figure 2) exploits *BJI_Workload*. It bases on closed frequent itemsets extraction technique. It involves three main steps:

- *Elaboration of the extraction context*: A syntaxical analysis of *BJI_Workload* must be conducted in this step to extract the set of attributes in selection and join predicates. We build after that a "query-attribute" matrix whose rows and columns represent respectively queries of *BJI_Workload* and the extracted attributes. The j^{th} case of row *i* in the matrix is set to 1 if the query Q_i uses the attribute A_j , and it is set to 0 otherwise.
- *Candidate indexes construction*: To prune the search space of the BJI selection problem, we use the algorithms: Close [17], Charm [25], DynaClose [5], DynaCharm [5]. The algorithms Close and Charm are using to extract closed frequent itemsets (CFIs), whereas DynaClose, DynaCharm present an adaptation of Close and Charm respectively. We eliminate from the set of CFIs those that cannot generate a bitmap join index. For example, a CFI which does not contain non-key attribute of a dimension tables will be deleted.
- *Index set selection*: We use, in this step, a greedy algorithm to scan the search space of candidate indexes under the storage space constraint S. The selection of the index set is based on the cost model proposed by Aouiche et al. in [1]. A candidate index *I*, will be added to the final BJI configuration if it minimizes the execution cost of *Q*.

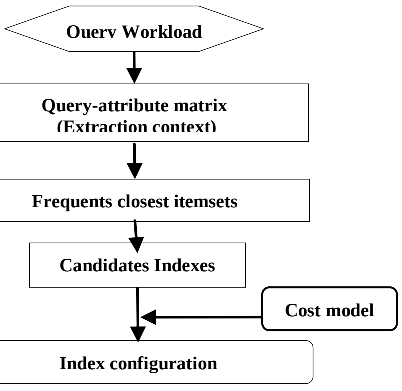

Figure2: BJI selection process

V. REFINING THE OPTIMIZATION

In order to improve the optimization, we propose as a refining step to select an additional BJI configuration *IC2* that optimizes the non-benefit queries of the partitioning schema *PS*. A query is called benefit of partitioning if his execution cost is significantly reduced after the partitioning. To quantify the cost reduction, we used a metric called reduction rate (R). This rate is calculated for a given query Q_j as follows:

$$R(Qj) = \frac{\text{Cost(Qj,BeforePartitioning)} - \text{Cost(Qj,AfterPartitioning)}}{\text{Cost(Qj,BeforePartitioning)}}$$

We gave the possibility to the administrator to fix a threshold $\lambda(\lambda \in [0,1])$, from which he considered a given query benefit of the partitioning process or not. It can be formulated as follows:

$$\begin{cases} R(Q_j) \geq \lambda & Q_j \text{ is a benefit query} \\ R(Q_j) < \lambda & Q_j \text{ is a non-benefit query} \end{cases}$$

It is very important to know if a query is benefit or not. When a query was a benefit query, the administrator considers that it is not necessary to apply an additional index selection to optimize here cost.

The selection of *IC2* must respect the space constraint S' where *S'= S-Size(IC)*.This refining step allows making the most of the storage space devoted to indexes, and thus ensuring better optimization of the overall workload.

VI. EXPERIMENTAL STUDY

To validate our approach, we have used a real data warehouse generated from the benchmark *APB-1 release II* [15] in Oracle11g. The star schema that we have identified from this benchmark consists of a fact table Actvars (24786 000 tuples), and four dimension tables: Prodlevel (9000 tuples), Custlevel (900 tuples), Timelevel (24 tuples) and Chanlevel (9 tuples). We used a query workload composed of 55 queries defined on 12 selection attributes. To show the effectiveness of our approach, we conducted the evaluation of the following four approaches: (a) without optimization techniques, which we call WOT,(b) HPOnly: only HP is used, (c) BJIOnly: only BJI are selected without partitioning and (d) HP-BJI: our approach. We compare these scenarios with the work proposed by Bouchakri et al. [6] (noted as HP-BJI-B). This approach is based on selection attributes classification. We used in this approach the GA for HP schema selection and a greedy algorithm to select the BJI set. In our experimental study, we have used the *Oracle Optimizer* to estimate the real execution cost of the workload. The Optimizer uses *Explain Plan* statement to estimate the execution cost. For example for a given query Q_i where:
Q_i: *Select * From Table Where Attribute = 'Value'*. It is possible to estimate the cost of execution of Q_i through the following two SQL commands:

- *EXPLAIN PLAN SET STATEMENT_ID = 'Q*$_i$*' FOR SELECT * FROM table WHERE Attribute = 'Value'*
- *SELECT Cost FROM plan_table WHERE STATEMENT_ID = 'Q*$_i$*'*

The first command displays the execution plan chosen by the optimizer to execute Q_i and then estimate the cost of execution of Q_i. This cost is stored in the *Cost* column of the system table *plan_table* and it is extracted by running the second command.

To validate our approach, we have conducted several experiments. In the first experiment, we varied W (maximal number of fact fragments) from 40 to 300. We used the algorithm Close [19] for BJI set selection because the experience shows that it presents the best results. For both HP-BJI and HP-BJI-B, we fix the storage space's threshold S to 1GB. Figures 3, 4 show respectively the results of these experiments and the cost reduction rate when executing the query workload for the approaches HPOnly, HP-BJI and HP-BJI-B.

The results indicated that our approach (HP-BJI) is more efficient than the other approaches. It is better than HP-BJI-B because of a large number of indexes that have been selected (7-9 BJI) for HP-BJI which represents more than half of the candidate indexes, while just (3-5) BJI are selected for HP-BJI-B.

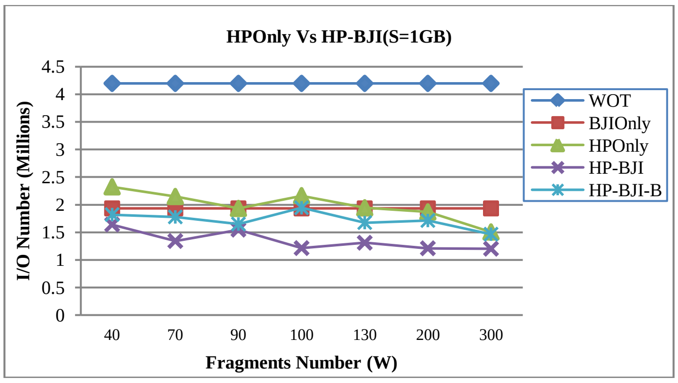

Figure 3: Comparison of HP-BJI and HPOnly.

This is due to the fact that HP-BJI-B does not take into account the indexes defined on the attributes used in the partitioning process.

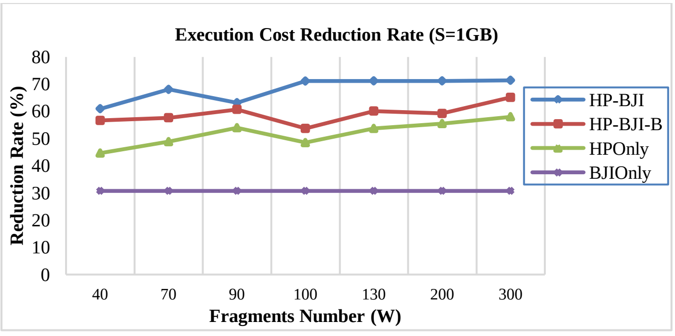

Figure 4: Cost reduction rate

The results in Figure 4 have shown that we get an average performance gain of more than 71% for our approach, which is considered a significant improvement.

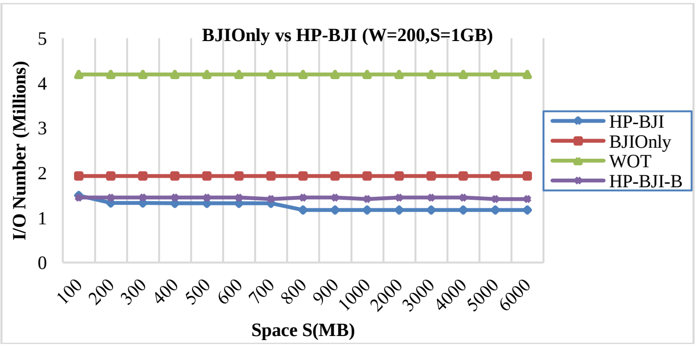

Figure 5: HP-BJI vs BJIOnly.

We compared, in the second experiment, our approach with BJIOnly and HP-BJI-B. We set W to 200 and we varied S from 100 to 6000MB.The results presented in Figure 5 show that our selection process HP-BJI is more efficient than HP-BJI-B and BJIOnly. For more than 800MB HP-BJI and HP-BJI-B are getting stable. This is due in the fact that the greedy algorithm cannot select other BJI which can present an improvement to the workload execution cost.

VII. CONCLUSION

We proposed in this paper a new approach based on data mining algorithms to reduce the complexity of HP and BJI combined selection's problem. The first data mining algorithm (*k-means*) is used to share the query workload into two sub-workloads. Each sub-workload is assigned to an optimization technique process (HP or BJI) according to classification criterions. The algorithms Close [17], Charm [25] to extract closed frequent itemsets with DynaClose [5] and DynaCharm [5] are implemented to be used for BJI configuration selection in order to optimize the global workload's execution cost. We used also a genetic algorithm for HP schema selection. This algorithm has been widely used in data mining applications such as classification, clustering, feature selection, association rules [23], etc. To validate our approach, we conducted several experiments on a benchmark on Oracle 11g. The experimental results are encouraging and have demonstrated the effectiveness of our approach. To improve this work, we think to propose a dynamic approach that adapts to recurrent changes in the workload, the data or the problem constraints to ensure continuous optimization.

REFERENCES

[1] Aouiche, K., Darmont, J., Boussaid, O., Bentayeb, F.: Automatic selection of bitmap join indexes in data warehouses. In: 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Lecture Notes in Computer Science, pp 64-73, Denmark (2005).

[2] Barr, M., Bellatreche, L.: A new approach based on ants for solving the problem of horizontal fragmentation in relational data warehouses. In: International Conference on Machine and Web Intelligence (2010).

[3] Bellatreche, L., Boukhalfa, K., Alimazighi, Z.: SimulPh.D.: A Physical Design Simulator Tool. In: 20th International Conference on Database and Expert Systems Applications (DEXA'09), LNCS, (2009).

[4] Bellatreche, L., Boukhalfa, K., Richard, P.: Data partitioning in data warehouses: Hardness study, heuristics and oracle validation. In: International Conference on Data Warehousing and Knowledge Discovery (DaWaK'2008), pp. 87-96 (2008).

[5] Bellatreche, L., Missaoui, R., Necir, H., Drias, H.: A data mining approach for selecting bitmap join indices. JCSE 1(2), pp. 177-194 (2007).

[6] Bouchakri, R., Bellatreche, L., Boukhalfa, K.: ODAG: Optimisation des requetes décisionnelles basée sur le Datamining et les algorithmes génétiques. Doctoriales STIC'11(2011).

[7] Boukhalfa, K., Bellatreche, L., Alimazighi Z.: HP&BJI: A combined selection of data partitioning and join indexes for improving OLAP performance. Annals of Information Systems, Special Issue on new trends in data warehousing and data analysis, pp. 179-2001. Springer (2008).

[8] Boukhalfa, K.: De la conception physique aux outils d'administration et de tuning des entrepôts de données. PhD thesis, University of Poitiers (2009).

[9] Chaudhuri, S.: Index selection for databases: A hardness study and a principled heuristic solution. IEEE Transactions on Knowledge and Data Engineering, vol. 16(11). pp. 1313-1323 (2004).

[10] Chaudhuri, S., Narasayya, V.: AutoAdmin 'What-if' Index Analysis Utility. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), pp. 367-378 (1998).

[11] Gacem, A., Boukhalfa K.: Very Large Workloads Based Approach to Efficiently Partition Data Warehouses. In: proceeding of the 4th International Conference on Computer Science and Its Applications (CIIA 2013). LNCS. Springer-Verlag (2013).

[12] Inmon, W. H.: Building the Data Warehouse. John Wiley (1992).

[13] Navarro, L.: Optimisation des bases de données Mise en œuvre sous Oracle. Pearson editor (2010).

[14] Necir, H.: A data mining approach for efficient selection bitmap join index. In: International Journal of Data Mining Modeling and Management vol. 2(3), pp. 238-251(2010).

[15] OLAP Council. Apb-1 Olap benchmark release II, http://www.olapcouncil.org/research/resrchly.html

[16] Ouared, A., Boukhalfa, k., Bellatreche L.: La Contribution des Agents pour la Conception Physique des Entrepôt de Données Relationnels. In: Second national conference (JEESI'12). Algeria(2012).

[17] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Proceedings of the 7th International Conference on Database Theory, ICDT '99, pp. 398-416, London (1999).

[18] Sanjay, A., Narasayya, V. R., Yang, B.: Integrating Vertical and Horizontal Partitioning Into Automated Physical Database Design. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 359-370 (2004).

[19] SPMF(sequential pattern mining framework). http://philippe-fournier-viger.com/spmf/.

[20] Stöhr T., Märtens H., Rahm E.: Multidimensional database allocation for parallel data warehouses. In: Proceedings of the International Conference on Very Large Databases, pp. 273-284 (2000).

[21] Tekaya, K., Abdelaziz, A., Habib, O.: Data mining based fragmentation technique for distributed data warehouses environment using predicate construction technique. Networked Computing and Advanced Information Management. pp. 63-68. (2010).

[22] Vanichayobon, S., Gruenwald, L.: Indexing techniques for data warehouses'queries. Technical report, University of Oklahoma, School of Computer Science (1999).

[23] Yan, X., Zhang, C., Zhang, S.: Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. Expert Syst. Appl. 36(2), pp. 3066-3076(2009).

[24] Yang, J., Karlapalem, K., Li, Q.: Algorithms for materialized view design in data warehousing environment. In; Proceedings of the International Conference on Very Large Databases, pp. 136-145 (1997).

[25] Zaki Mohammed, J., Hsiao, C.J.: Charm : An efficient algorithm for closed itemset mining. In: Proceedings of the Second SIAM International Conference on Data Mining, pp. 457-473, Arlington (2002).

[26] Ziani, B., Ouinten, Y.: Vers l'auto-sélection des index dans les entrepôts de données : une approche basée sur la recherche des motifs fréquents maximaux. CARI2010. pp. 301-308, Yamoussoukro (2010).

[27] Ziani, B., Rioult, F., Ouinten, Y.: A constraint based mining approach for multiattribute index selection. In: 14th International Conference on Enterprise Information Systems, pp. 93-98. SciTePress (2012).

[28] Ziani, B., Benmlouka, A., Ouinten, Y.: Improving Index Selection Accuracy for Star Join Queries Processing: An Association Rules Based Approach. Advances in Intelligent Systems and Computing, vol. 220, pp 67-74 (2013).

[29] Zilio, C., Rao, J., Lightstone, S., Lohman, G., Storm, A., Garcia-Arellano C., and Fadden S.: DB2 design advisor: integrated automatic physical database design. In Proceedings of the Thirtieth international conference on very large data bases (VLDB '04), pp. 1087-1097 (2004).