OPTASSIST: A RELATIONAL DATA WAREHOUSE OPTIMIZATION ADVISOR

KAMEL BOUKHALFA[1], ZIANI BENAMEUR[3], LADJEL BELLATRECHE[2], ZAIA ALIMAZIGHI [1]

[1] USTHB-Algiers, Algeria	boukhalk@ensma.fr
[2] LISI/ENSMA Poitiers-France	bellatreche@ensma.fr
[3] Laghouat University – Algeria	bziani@mail.lagh-univ.dz

ABSTRACT

Data warehouses store large amounts of data usually accessed by complex decision making queries with many selection, join and aggregation operations. To optimize the performance of the data warehouse, the administrator has to make a physical design. During physical design phase, the Data Warehouse Administrator has to select some optimization techniques to speed up queries. He must make many choices as optimization techniques to perform, their selection algorithms, parameters of these algorithms and the attributes and tables used by some of these techniques. We describe in this paper the nature of the difficulties encountered by the administrator during physical design. We subsequently present a tool which helps the administrator to make the right choices for optimization. We demonstrate the interactive use of this tool using a relational data warehouse created and populated from the APB-1 Benchmark.

KEY WORDS*:* Optimization, Data Warehouse, physical design, horizontal partitioning, Bitmap join index

1 INTRODUCTION

The main characteristics of data warehouses are their large size and complexity of OLAP (On-Line Analytical Processing) queries due to the operations of selection, join and aggregation. These characteristics have made the task of administering increasingly complex. Traditionally, in databases business applications like OLTP (On-Line Transaction Processing), the task of an administrator was mainly concentrated on the user management and use of a limited number of optimization techniques as indexes and views.

Optimizing the execution time of queries is a key requirement of data warehouse users. To satisfy this requirement, the data warehouse administrator (DWA) must perform a physical design that is crucial to ensure good performance. The physical design must determinate how a query should be run efficiently on the data warehouse. Thus, the DWA has a set of optimization techniques such as vertical partitioning, horizontal partitioning, indexes, etc. He can use a single technique or combine several to get a better performance. Several selection algorithms are available for a given technique. Each algorithm is characterized by a set of parameters to adjust. For some techniques, several objects from the data warehouse are candidates (usually tables and attributes). To conduct the physical design task, the DWA faces make several choices related to: (1) optimization techniques, (2) selection mode, (3) selection algorithms and their parameters, and (4) tables and attributes candidates (Figure 1):

1.1 Choice of optimization techniques

If we explore the literature and commercial DBMS, we find a wide variety of optimization techniques that may be redundant or not. Redundant techniques require storage space and maintenance costs (materialized views, advanced indexes, vertical partitioning, etc.), while non-redundant techniques do not require storage or maintenance costs (horizontal fragmentation, parallel processing, etc.). To optimize queries defined on the data warehouse, the DWA can choose one or more optimization techniques among these two categories. This choice is often difficult because some techniques are beneficial for some queries and not for others.

1.2 Choice of selection mode

With several optimization techniques, the DWA has two modes of selection: isolated selection and multiple selection. In isolated selection, he chose one technique that may be redundant if it has enough space and few updates, for example, or he may choose a non-redundant technique. The isolated selection has been widely studied [4,5,9,13,10], but it is often insufficient for a better optimization of the data warehouse. Multiple selection consist to select several techniques at once. It is mainly motivated by the strong similarities between optimization techniques. The major works in this category are mainly concentrated on the selection of materialized views and indexes [3,16,17]

1.3 Choice and setting of selection algorithms

Once the optimization techniques used are chosen, the DWA faces the problem of choice of their selection algorithms. For each selection mode, isolated or multiple, a wide choice of algorithms is possible. These algorithms are of various types, ranging from simple algorithms such as greedy algorithms to more complex algorithms such as algorithms based on linear programming, genetic algorithms, ant colonies, etc. Some algorithms have few parameters such as greedy algorithms while others have several parameters that must be configured for good performance.

1.4 Choice of candidate attributes and tables

The relational data warehouses are generally modeled by a star schema consists of a fact table and a set of dimension tables. For some techniques such as horizontal partitioning (HP), vertical partitioning and indexes, several tables and attributes are candidates to be used by these techniques. The DWA must choose in some cases a subset of tables and attributes among the initial set. This choice is often made to prune search space and reduce the complexity of the selection problem.

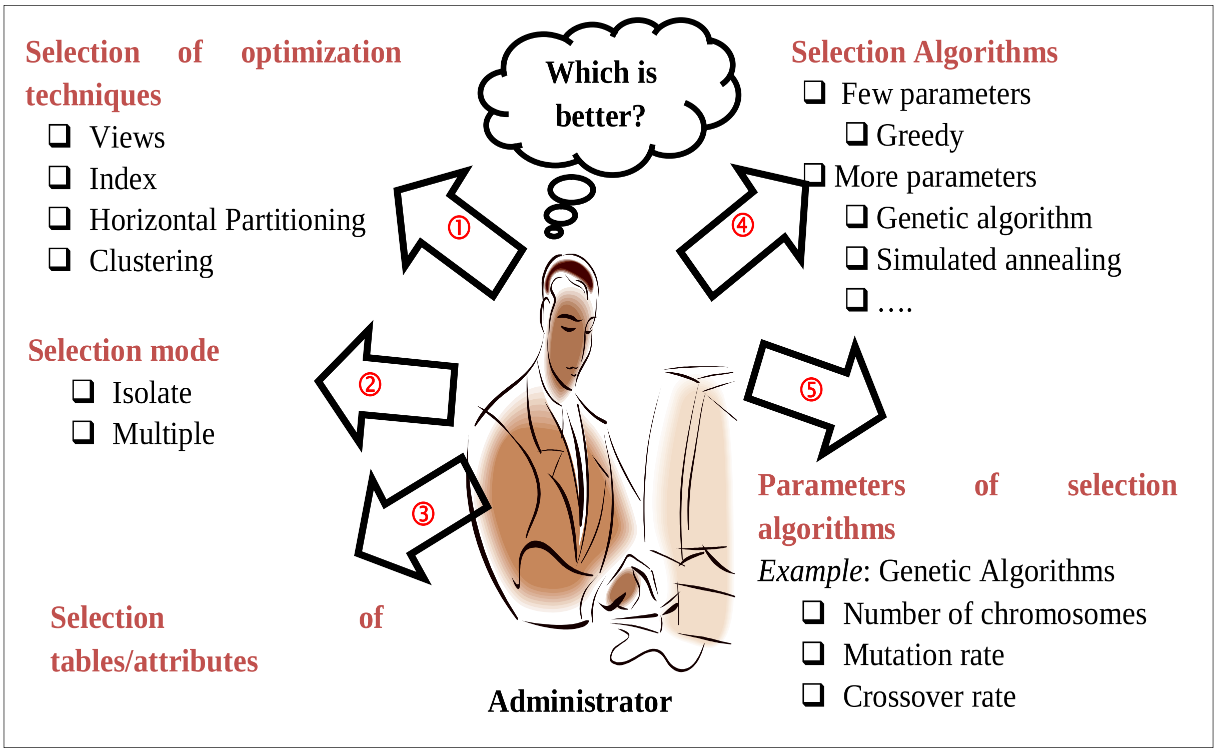


Figure 1: Choices made by administrator

As we have seen, the task of the DWA is becoming increasingly complex given the large number of choices to make, hence the need for development of advisor tools. These tools should assist the DWA to make the right choices for data warehouse optimization.

This paper is organized into 6 sections. Section 2 presents a state of the art on the advisor tools developed to assist the administrator in his optimization task. Section 3 presents the optimization techniques discussed in this paper and their interaction. Section 4 presents the general architecture of the tool that we develop. Section 5 is devoted to the presentation of our tool features applied to a data warehouse. Finally, Section 6 concludes the paper and presents some perspectives.

2 STATE OF THE ART

To assist the DWA in optimizing the data warehouse some tools have been developed. Most existing tools have been proposed by commercial DBMS in the context of self-administration. Among these tools, we can cite *Oracle SQL Access Advisor* [1], *DB2 Design Advisor* [18] and *Microsoft Database Tuning Advisor* [2].

SQL Access Advisor provides comprehensive advice on how to optimize the design of a scheme to maximize application performance. This tool is a wizard that automates some aspects of physical design and tuning[1] performed manually on Oracle databases. The tool analyzes the workload of queries and offers recommendations for creating new index if necessary, remove unused indexes, create new materialized views, etc. The recommendations generated are accompanied by a quantified assessment of the performance gains and scripts necessary to implement them.

DB2 Design Advisor is a part of *DB2 V8.2*. It is an improvement of *DB2 Index Advisor Tool* which selects a set of indexes. *DB2 Design Advisor* optimizes a set of queries by proposing a set of recommendations. These recommendations concern four optimization techniques: indexes, materialized views, HP and clustering.

Microsoft Database Tuning Advisor (DTA) is developed in *Microsoft Research AutoAdmin project*. DTA can provide integrated recommendations for indexes defined on tables, views, indexes defined on views and horizontal partitioning. It takes as input a set of data bases on a server, a workload of queries, optimization techniques to select and a set of constraints, such as storage cost for redundant techniques. It will output a set of recommendations for indexes, views and HP.

Most of the tools that we have presented have been proposed in the context of self-administration of databases and are generally specific to a given DBMS. They are also characterized by the use of the query optimizer to evaluate the quality of selected techniques. This represents an additional task of the optimizer and can cause a deterioration of performance.

In trying to automate the administration and tuning of databases and data warehouses, the authors of these tools seek to discharge the administrator of these two tasks. [12] Shows that a physical design developed without administrator intervention poses a problem of robustness. Optimization techniques generated can degrade performance instead of improving them. The algorithms used by these tools for the selection of optimization techniques are frozen and not accessible to the administrator. It is interesting to improve this toolkit by other tools for allowing more interactivity with the administrator. These tools should allow the administrator to customize his physical design and use his experience to improve the quality of selected optimization techniques. In this way, we propose *OptAssist* a tool for assisting the DWA in his data warehouse optimization task.

By using OptAssist the DWA can choose optimization techniques, selection mode, used algorithms, parameters for each algorithm as well as tables and attributes considered for the generation of recommendations. Unlike most tools which provide only primary partitioning, *OptAssist* can recommend a primary and derived horizontal partitioning. It also allows multiple selections of HP and bitmap join indexes (BJI) to better optimize the data warehouse. The tool uses a cost model that we proposed in [8]. It supports multiple DBMS by exploiting the meta-base to collect all information and statistics needed for optimization.

3 THE USED OPTIMIZATION TECHNIQUES

Due to the large number of existing optimization techniques, OptAssist concentrates on three optimization techniques: (1) primary horizontal partitioning, (2) derived horizontal partitioning (both are non redundant techniques), and (3) bitmap join indexes (redundant technique). This choice is performed due to similarities between them. Primary horizontal partitioning allows a table to be decomposed into disjoint sets of rows using selection attributes of that table. This partitioning can supported using several modes: Range, List, Hash, Composite, etc. [15]. The derived partitioning allows the decomposition of a table based on attributes of another partitioned table(s). The derived partitioning of a table R based on the fragmentation schema of S is feasible, if and only if, there is a join link between R and S (R contains a foreigner key of S). It is similar to referential partitioning recently supported by ORACLE11G. Most of today's commercial DBMSs include a DDL (data definition language) support for defining horizontal partitions of a table. Primary and

[1]*The tuning is a set of activities used to optimize the performance of a database or data warehouse as a result of their evolution.*

derived partitioning optimizes selections (prune partition) and joins operations (partition-wise joins).

Example 1: For example, let dimension table *TimeLevel* be a list partitioned table using QuarterLevel attribute created as follows:

CREATE TABLE TimeLevel(Tid VARCHAR2(12) NOT NULL, Year_Level NUMBER, Quarter_Level VARCHAR2(2), Month_Level NUMBER, PRIMARY KEY (Tid))

PARTITION BY LIST(Quarter_Level)

(PARTITION First_Quarter VALUES('Q1') TABLESPACE TL_TB1,

PARTITION Second_Quarter VALUES('Q2') TABLESPACE TL_TB2,

PARTITION Third_Quarter VALUES('Q3') TABLESPACE TL_TB3,

PARTITION Fourth_Quarter VALUES('Q4') TABLESPACE TL_TB4)

Then users can derive partition the fact table Actvars into 4 fragments using the referential partitioning supported by ORACLE11G as follows:

CREATE TABLE Actvars(Customer_Level VARCHAR2(12) NOT NULL, Product_Level VARCHAR2(12) NOT NULL, Channel_Level VARCHAR2(12) NOT NULL, Time_Level VARCHAR2(12) NOT NULL, Unitssold FLOAT, Dollarsales FLOAT, Dollarcost FLOAT,

CONSTRAINT fk_Actvars_TimeLevel FOREIGN KEY (Time_Level) REFERENCES TimeLevel(Tid))

PARTITION BY REFERENCE (fk_Actvars_TimeLevel) Bitmap join index is proposed to speed up join and selection operations. In its simplest form, it can be defined as a bitmap index on a table R based on column(s) of another table S, where S commonly joins with R in a specific way.

Example 2: A bitmap join index between the fact table Actvars and dimension table TimeLevel based on the attribute Quarter Level is defined as follows:

CREATE BITMAP INDEX Quarter_TimeLevel_Actvars_bjix

ON Actvars (TimeLevel.Quarter_Level)

FROM Actvars A, TimeLevel T

Where A.Time_level=T.Tid

Based on the two examples, we can easily identify similarities between horizontal partitioning and bitmap join indexes that we describe in the following section.

In [18], the authors argued that primary horizontal partitioning and single table indexes weakly depend on each other. Their argumentation is based on the fact that complex queries tend to use hash joins more often. In their study, they did not consider multiple table indexes, such as BJI. For OLAP queries (such as COUNT(*) queries), these indexes are usually used instead of hash joins. We claim that derived horizontal partitioning and BJI are two dependent techniques - both optimize joins and selections and usually compete for the same resource representing selection attributes defined in queries. A major difference between these techniques is that BJI are suitable for selection attributes with a low cardinality such as Gender (called indexable attributes), but horizontal partitioning can be generated using any selection attribute (called fragmentation attributes). To show the nature of these interdependencies, we consider the following scenarios. Let FAC and IAC be the set of fragmentation and indexation attributes candidate for partitioning and indexing processes, respectively.

- If ($FAC \cap IAC = \varphi$), derived horizontal partitioning and bitmap join indexes weakly depend on each other, since they do not compete for the same selection attributes. DWA may partition the data warehouse using FAC and then select BJI on the fragmented schema using IAC.
- If ($FAC \cap IAC \neq \varphi$) two cases may possible (i) the DWA could consider that these two techniques weakly depend on each other. This choice is not interesting for the following complexity reasons: selecting a fragmentation schema of a relational data warehouse is a hard problem [4]. Its complexity is proportional to the number of fragmentation attributes candidate [14]. The same remark goes for selecting BJI [7]. (ii) Instead of selecting these two techniques sequentially, it is better to select them using combined selection mode as follows: selecting derived horizontal partitioning using FAC and selecting BJI on ($IAC-FA$) set, where FA is the set of attributes participating in fragmenting the data warehouse. In this case, BJI strongly depend on horizontal partitioning.

Our optimization tool should take into account these interdependencies and gives DWA choices on choosing his favorite mode.

4 ARCHITECTURE OF OPTASSIST

OptAssist accepts as input a data warehouse schema, a workload of queries Q and a set of constraints (the maximum number of fragments, W, for HP and the quota of storage space, S, for BJI). It can fragment or indexing the data warehouse or both operations simultaneously. Our choice to use the HP and BJI is motivated by several similarities that we have identified between these two techniques in [8]. OptAssist consists of a set of modules assisting the DWA to make his optimization choices (see Figure 2): (1) meta-base querying module, (2) managing queries module, (3) HP selection module, (4) BJI selection module, (5) horizontal partitioning module, (6) indexing module and (7) query rewriting module.

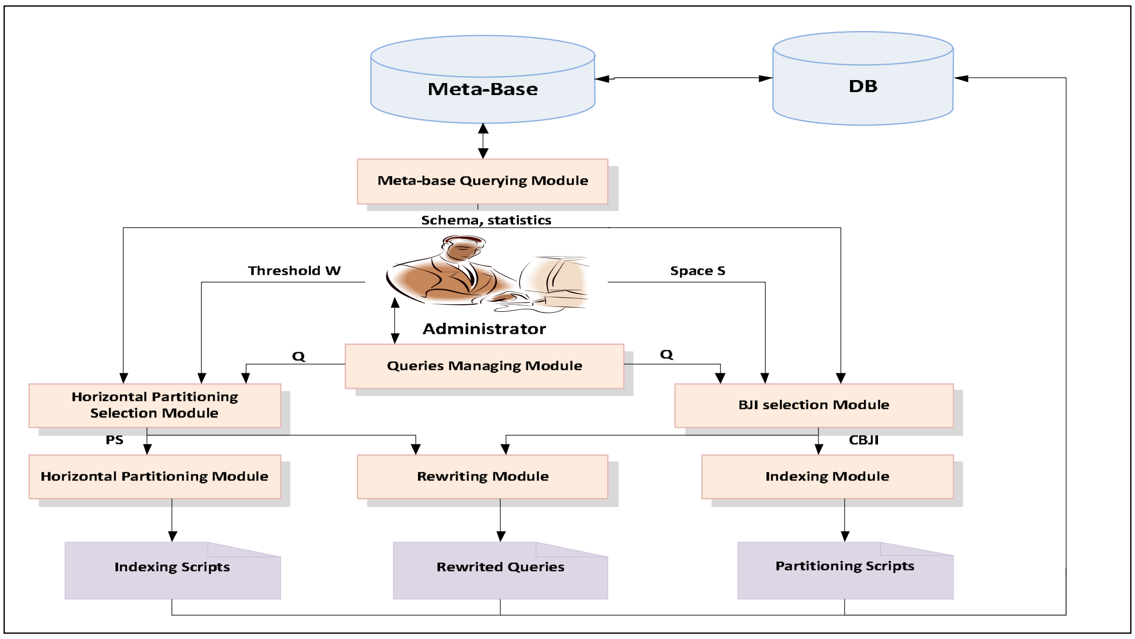


Figure 1: OptAssist architecture

4.1 Meta-base Querying Module

The meta-base querying module is a very important module that allows the tool to work with any type of DBMS. From a type of DBMS, user name and password the module allows to connect to that (an) account and collect some information from the meta-base. These information concern logical and physical levels of the data warehouse. The information of the logical level includes tables and attributes in these tables. The information of the physical level includes optimization techniques used and a set of statistics on tables and attributes of the data warehouse (number of tuples, cardinaly, etc.).

4.2 Managing Queries Module

This module enables the DWA to help define the workload of queries (Q) on which the optimization is based. The module allows manual editing of a query or import from external files. It may also manage the workload, giving the possibility to add, delete or update queries. This module integrates *parser* that identifies syntax errors as well as tables and attributes used by each query.

4.3 Horizontal Partitioning Selection Module (HPSM)

HPSM requires as input a schema of data warehouse, a workload and a threshold W representing the maximum number of fragments that the administrator can manage. Using these data, HPSM selects a partitioning schema (PS) to minimize the cost of the workload and generating a number of fragments not exceeding W. In [4], we conducted a complexity study of the HP selection problem in the context of relational data warehouses, and we proved that it is NP-complete. Therefore, to find a solution to this problem, we proposed three heuristic algorithms: Genetic Algorithm (GA), Simulated Annealing algorithm (SA) and Hill Climbing algorithm (HC) [4,6]. These three algorithms are supported in the HPSM.

4.4 BJI Selection Module

This module requires as input a schema of the data warehouse, a workload Q and storage space S allocated to BJI. It selects a configuration of BJI (*CBJI*) to minimize the workload execution time respecting the constraint Size(CBJI) <= S. The module supports two greedy algorithms that we proposed in [8] and an algorithm based on a technique of data mining (Closed frequent Itemsets) proposed by Aouiche et al. [3].

4.5 Horizontal Partitioning Module (HPM)

HPM fragments physically the data warehouse using partitioning schema obtained from HPSM. From the partitioning schema, HPM determines the dimension tables to fragment by horizontal primary partitioning and attributes used to perform this fragmentation. The module can then fragment the fact table by horizontal derived partitioning using fragments of dimension tables. In [8] we identified two problems: (1) most DBMS do not support the primary HP on three attributes or more and (2) derived HP is not supported in the case of two dimension tables or more. We have proposed a technique to solve these two problems. This technique is supported by the HPM. This generates all scripts that allow partitioning fact and dimension table as the input partitioning schema PS.

4.6 Indexing Module

The indexing module is responsible for the creation of BJI selected by the BJI selection module. This module generates SQL queries to create BJI on the data warehouse.

4.7 Query Rewriting Module

Once the optimization techniques physically created on the data warehouse (HP and/or BJI), a step of rewriting queries is necessary. Two types of rewrites are performed: rewriting for BJI and rewriting for HP. Rewriting for BJI is to add Hints in the SELECT clause of queries to force the use of created BJI[2]. Rewriting for HP is to identify valid fragments for each query, rewrite the query on each of these fragments and finally performing union of the obtained results.

[2] *The hints INDEX in a query force using one or more indexes to execute this query.*

5 FEATURES OF OPTASSIST

We present in this section the main features of OptAssist through its use on a real data warehouse generated from the APB-1 Benchmark [11]. The star schema that we have reached from this Benchmark consists of a fact table *Actvars* (24 786 000 tuples) and four dimension tables, *Prodlevel* (9 000 tuples), *Custlevel* (900 tuples), *Timelevel* (24 tuples) and *Chanlevel* (9 tuples).

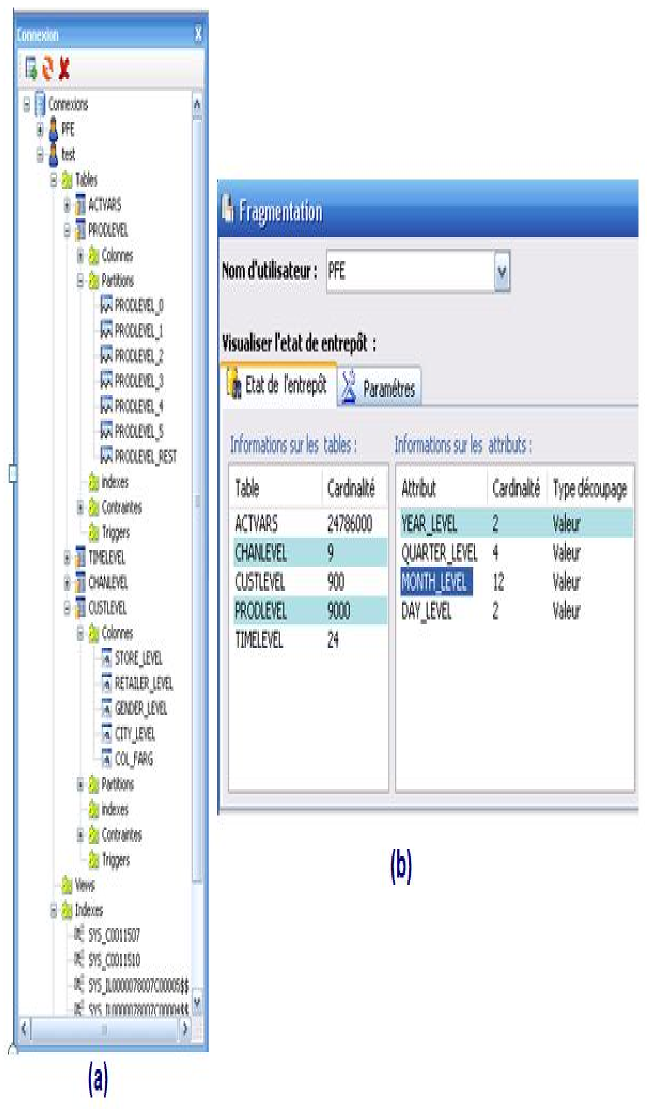
To assist the DWA in optimization of the data warehouse, OptAssist performs four main functions: visualization of data warehouse current status, preparing optimization, partitioning the data warehouse and indexing the data warehouse (fragmented or not). We present below the four features.

5.1 Visualization of data warehouse current status

Displaying the status of the data warehouse allows DWA to know the data warehouse schema, dimension tables, fact table and some statistics on these tables. OptAssiste can also display optimization techniques already created on the data warehouse. All this information is collected through the meta-base querying module. This visualization allows the administrator to have an overall view of his data warehouse before beginning the optimization process. Figure 3 (a) shows an example of visualization where tables, attributes and optimization techniques created are displayed. Figure 3 (b) shows a set of statistics collected about some objects in the data warehouse.

5.2 Preparing the optimization

The preparation of optimization is to collect the information necessary to perform this optimization. It concerns the preparation of the work load *Q*, the choice of selection mode and definition of some physical parameters. Figure 4 shows the interface of managing the workload of queries where DWA can add, edit or delete a query and check its syntax. OptAssist supports two modes of selection: isolated and multiple. In the isolated mode, the DWA can use horizontal partitioning only (HPONLY) or BJI only (BJIONLY) to optimize his data warehouse. Multiple selection mode is to use both techniques HP and BJI. First, the data warehouse is fragmented into a set of fragments, then these fragments are indexed. OptAssist allows the DWA to fixe some physical parameters such as buffer size and page system size.



```
Connexions
  PFE
  test
    Tables
      ACTVARS
      PRODLEVEL
        Colonnes
        Partitions
          PRODLEVEL_0
          PRODLEVEL_1
          PRODLEVEL_2
          PRODLEVEL_3
          PRODLEVEL_4
          PRODLEVEL_5
          PRODLEVEL_REST
        Indexes
        Contraintes
        Triggers
      TIMELEVEL
      CHANLEVEL
      CUSTLEVEL
        Colonnes
          STORE_LEVEL
          RETAILER_LEVEL
          GENDER_LEVEL
          CITY_LEVEL
          COL_FARG
        Partitions
        Indexes
        Contraintes
        Triggers
    Views
    Indexes
      SYS_C0011507
      SYS_C0011510
      SYS_IL0000078007C00005$$
      SYS_IL0000078007C00004$$
```

Nom d'utilisateur : PFE

Visualiser l'etat de entrepôt :

Etat de l'entrepôt | Paramètres

Informations sur les tables :

Table	Cardinalité
ACTVARS	24786000
CHANLEVEL	9
CUSTLEVEL	900
PRODLEVEL	9000
TIMELEVEL	24

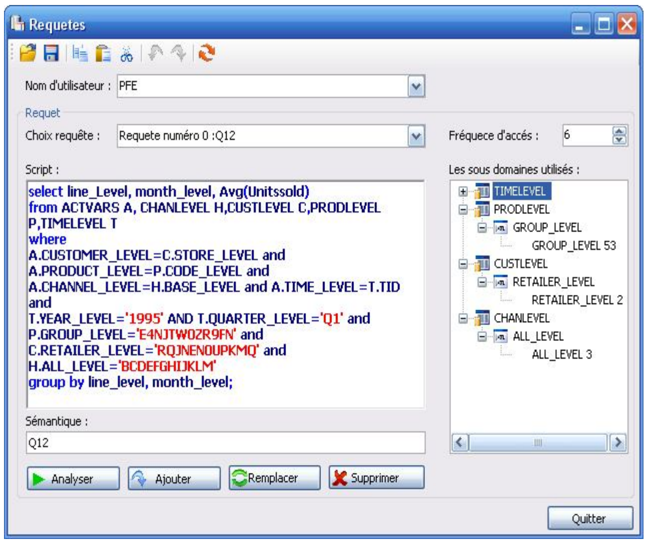
Informations sur les attributs :

Attribut	Cardinalité	Type découpage
YEAR_LEVEL	2	Valeur
QUARTER_LEVEL	4	Valeur
MONTH_LEVEL	12	Valeur
DAY_LEVEL	2	Valeur

Figure 3: (a) Visualization of data warehouse state (b) Visualization of statistics

5.3 The partitioning of the data warehouse

The partitioning of the warehouse is to fragment the dimension tables by the primary HP and the fact table by derived HP. The DWA begins by choosing the maximum number of fragments (W) then chooses whether he wants a personalized fragmentation or not. If he chooses non-personalized fragmentation then OptAssist partitions the data warehouse using all candidate attributes and tables and a default partitioning algorithm. Personalized partitioning offers more options to DWA in the selection process. He can choose dimension tables and attributes involved in the partitioning process. He must choose the partitioning algorithm (GA, SA or HC) and set its parameters. Figure 5 represents the algorithms choice and setting interface. For each selected algorithm, *OptAssist* activates the corresponding parameters and allows the DWA to change theses parameters. To illustrate the personalized partitioning we consider the case where the EDA chose to eliminate specific attributes and tables in the process of fragmentation. Figure 6 and 7 represent respectively the non-personalized and personalized partitioning interfaces. If the DWA chose to personalize the partitioning process, then *OptAssist* gives it the possibility to choose candidates dimension tables and attributes. In Figure 7, the DWA has eliminated the table *CustLevel*, one attribute of the table *TimeLevel* and three attributes of the table *ProdLevel* from the partitioning process. After selecting tables and attributes, the DWA selects the simulated annealing algorithm, set W to 100 and starts the selection. The partitioning schema obtained with this personalization generates 72 fragments with about 8,8 % of cost reduction compared to non-personalized partitioning.



Nom d'utilisateur : PFE

Choix requête : Requete numéro 0 :Q12

Fréquece d'accés : 6

Script :

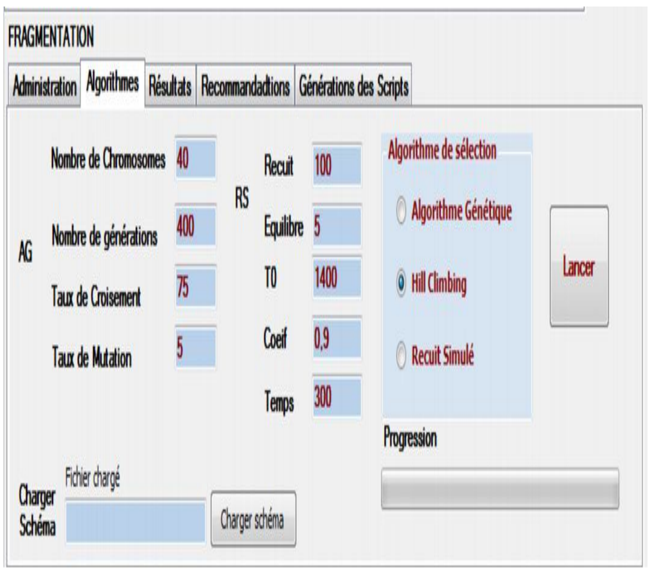
```
select line_Level, month_level, Avg(Unitssold)
from ACTVARS A, CHANLEVEL H,CUSTLEVEL C,PRODLEVEL
P,TIMELEVEL T
where
A.CUSTOMER_LEVEL=C.STORE_LEVEL and
A.PRODUCT_LEVEL=P.CODE_LEVEL and
A.CHANNEL_LEVEL=H.BASE_LEVEL and A.TIME_LEVEL=T.TID
and
T.YEAR_LEVEL='1995' AND T.QUARTER_LEVEL='Q1' and
P.GROUP_LEVEL='E4NJTWOZR9FN' and
C.RETAILER_LEVEL='RQJNENOUPKMQ' and
H.ALL_LEVEL='BCDEFGHIJKLM'
group by line_level, month_level;
```

Les sous domaines utilisés :

```
TIMELEVEL
PRODLEVEL
  GROUP_LEVEL
    GROUP_LEVEL 53
CUSTLEVEL
  RETAILER_LEVEL
    RETAILER_LEVEL 2
CHANLEVEL
  ALL_LEVEL
    ALL_LEVEL 3
```

Sémantique : Q12

Figure 4: Workload managing interface

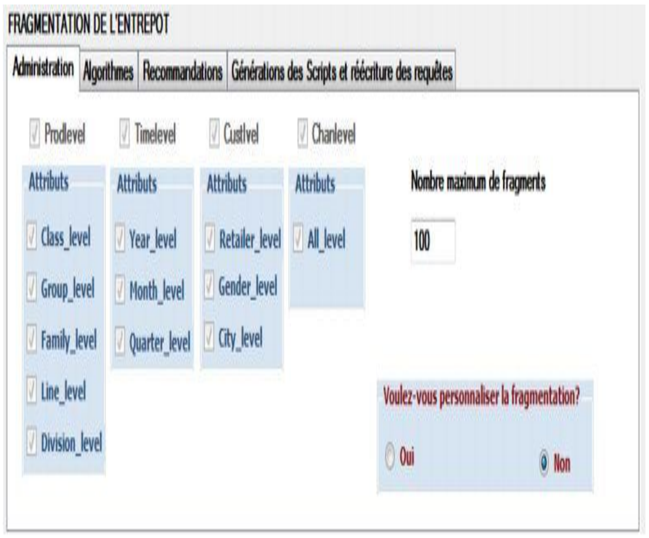


AG	Nombre de Chromosomes	40	
	Nombre de générations	400	
	Taux de Croisement	75	
	Taux de Mutation	5	
RS	Recuit	100	
	Equilibre	5	
	T0	1400	
	Coeif	0,9	
	Temps	300	

Algorithme de sélection: Algorithme Génétique [✓ Hill Climbing] Recuit Simulé

Charger Schéma — Fichier chargé

Figure 5: Algorithms selection and setting



FRAGMENTATION DE L'ENTREPOT

Administration | Algorithmes | Recommandations | Générations des Scripts et réécriture des requêtes

✓ Prodlevel | ✓ Timelevel | ✓ Custlevel | ✓ Chanlevel

Attributs	Attributs	Attributs	Attributs
✓ Class_level	✓ Year_level	✓ Retailer_level	✓ All_level
✓ Group_level	✓ Month_level	✓ Gender_level	
✓ Family_level	✓ Quarter_level	✓ City_level	
✓ Line_level			
✓ Division_level			

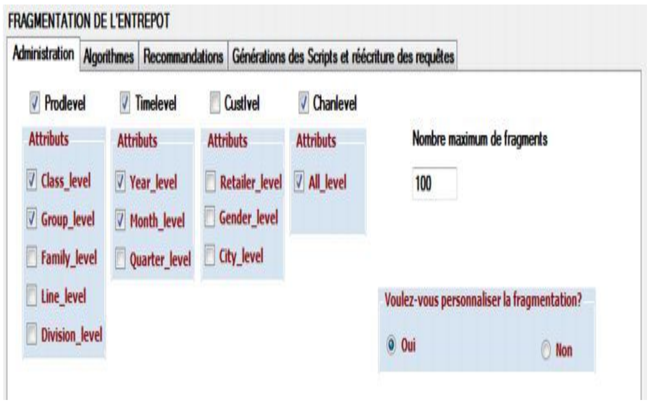
Nombre maximum de fragments: 100

Voulez-vous personnaliser la fragmentation?

Oui [✓ Non]

Figure 6: Non personalized partitioning

Once the partitioning schema of the data warehouse is selected by the HPSM, the DWA may visualize a recommendation proposed by OpAssist. This recommendation provide more information: number of generated fragments, fragmented dimension tables, attributes used to partition these tables, an estimated number of input-output necessary to execute the workload, the number of fragments of each dimension table, the gain performance obtained by fragmentation (compared to unfragmented schema), etc. Figure 8 shows the interface displaying attributes used to partition the data warehouse (four attributes among twelve were used: Line _level, Year_level, Month_level and All _level). If the DWA is not satisfied with this recommendation, he may return to the previous steps and change the various settings (reselect attributes and tables, or algorithms, parameters, etc..). The back is essential in the physical design phase in order to improve optimization. When the DWA is satisfied, he may ask OptAssist to generate fragmentation scripts and rewrite queries.



FRAGMENTATION DE L'ENTREPOT

Administration | Algorithmes | Recommandations | Générations des Scripts et réécriture des requêtes

✓ Prodlevel | ✓ Timelevel | Custlevel | ✓ Chanlevel

Attributs	Attributs	Attributs	Attributs
✓ Class_level	✓ Year_level	Retailer_level	✓ All_level
✓ Group_level	✓ Month_level	Gender_level	
Family_level	Quarter_level	City_level	
Line_level			
Division_level			

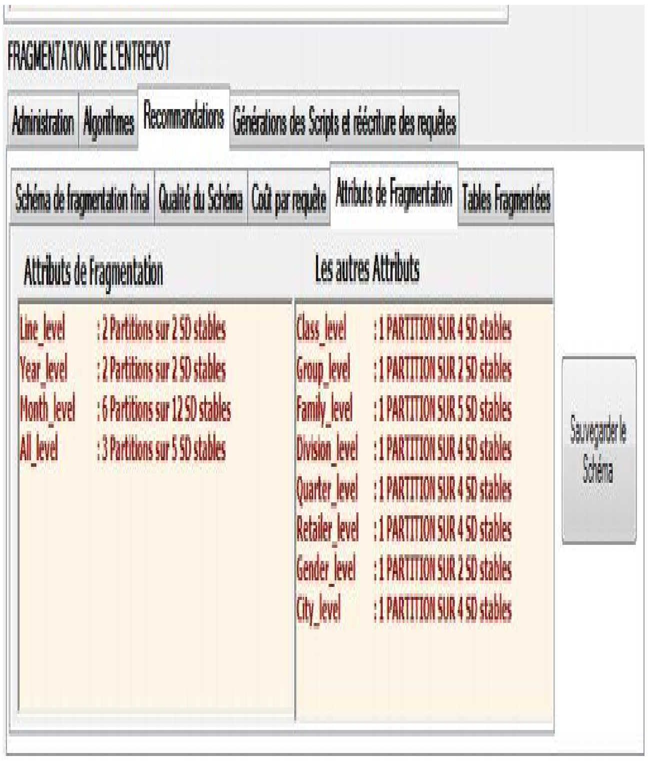
Nombre maximum de fragments: 100

Voulez-vous personnaliser la fragmentation?

[✓ Oui] Non

Figure 7: Personalization of partitioning

To physically fragmenting the data warehouse, the DWA executes the generated scripts; the unfragmented data warehouse will be replaced by the fragmented data warehouse.



FRAGMENTATION DE L'ENTREPOT

Administration | Algorithmes | Recommandations | Générations des Scripts et réécriture des requêtes

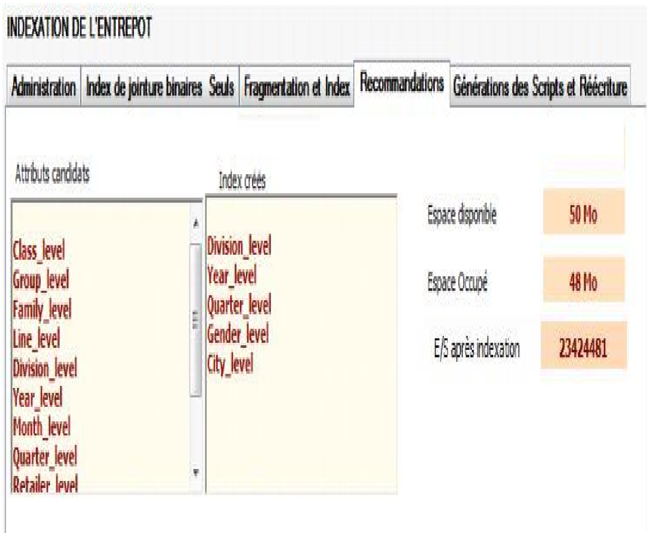
Schéma de fragmentation final | Qualité du Schéma | Coût par requête | Attributs de Fragmentation | Tables Fragmentées

Attributs de Fragmentation		Les autres Attributs	
Line_level	: 2 Partitions sur 2 SD stables	Class_level	: 1 PARTITION SUR 4 SD stables
Year_level	: 2 Partitions sur 2 SD stables	Group_level	: 1 PARTITION SUR 2 SD stables
Month_level	: 6 Partitions sur 12 SD stables	Family_level	: 1 PARTITION SUR 5 SD stables
All_level	: 3 Partitions sur 5 SD stables	Division_level	: 1 PARTITION SUR 4 SD stables
		Quarter_level	: 1 PARTITION SUR 4 SD stables
		Retailer_level	: 1 PARTITION SUR 4 SD stables
		Gender_level	: 1 PARTITION SUR 2 SD stables
		City_level	: 1 PARTITION SUR 4 SD stables

Figure 8: Attributes used to partition the data warehouse

5.4 Indexing the warehouse

OptAssist supports two modes of indexing the data warehouse: isolated (ONLYBJI) and multiple (HP&BJI). In the case of ONLYBJI mode, the DWA must first choose the candidate indexable attributes and storage space S. As with HP, two types of indexing are possible: non-personalized indexing and personalized indexing. The BJI selection module supports three selection algorithms, two greedy algorithms (one for selecting single attribute BJI and the second for selecting multiple attributes BJI) and a data mining based algorithm. OptAssist generates a recommendation that provides some information: BJI selected, percentage of cost reduction, indexed tables and attributes, storage cost, etc. To illustrate this, we believe that the DWA chose to make a non-personalized indexing with a space storage of 50 MB. Figure 9 show interface dedicated to the recommendations generated after BJI selection. Among the twelve indexable attributes, five attributes have been used to create five BJI occupying 48 MB.


Administration
Index de jointure binaires
Seuls
Fragmentation et Index
Recommandations
Générations des Scripts et Réécriture

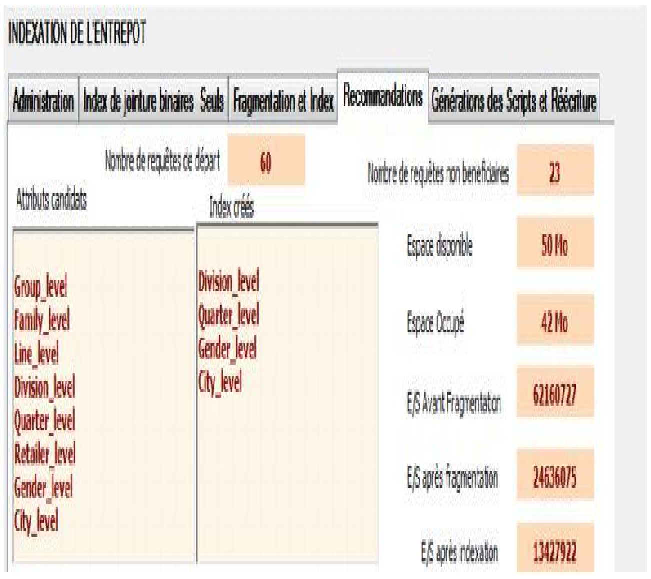
INDEXATION DE L'ENTREPOT

Attributs candidats: Class_level, Group_level, Family_level, Line_level, Division_level, Year_level, Month_level, Quarter_level, Retailer_level

Index créés: Division_level, Year_level, Quarter_level, Gender_level, City_level

Espace disponible	50 Mo
Espace Occupé	48 Mo
E/S après indexation	23424481

Figure 9: BJI recommendations in ONLYBJI selection mode


Administration
Index de jointure binaires
Seuls
Fragmentation et Index
Recommandations
Générations des Scripts et Réécriture

INDEXATION DE L'ENTREPOT

Nombre de requêtes de départ: 60

Nombre de requêtes non beneficiaires: 23

Attributs candidats: Group_level, Family_level, Line_level, Division_level, Quarter_level, Retailer_level, Gender_level, City_level

Index créés: Division_level, Quarter_level, Gender_level, City_level

Espace disponible	50 Mo
Espace Occupé	42 Mo
E/S Avant Fragmentation	62160727
E/S après fragmentation	24636075
E/S après indexation	13427922

Figure 10: BJI recommendations in HP&BJI selection mode

Indexing in FH&BJI mode is to index the partitioned data warehouse. The difference between ONLYBJI and FH&BJI indexation modes is the choice of candidate indexable attributes and the set of queries used to optimize the data warehouse. In ONLYBJI mode, all candidates indexable attributes and all queries are used by the selection algorithms. In FH&BJI mode, candidate indexable attributes are chosen among indexable attributes unused to partition the data warehouse. The queries used by selection algorithms in this mode are all queries that do not benefit from partitioning. To illustrate this indexing mode, consider that the DWA seeks to index the partitioned data warehouse. After partitioning, OptAssist disables automatically attributes used to partition the data warehouse, since they are not used to index the data warehouse. The DWA chooses the greedy algorithm, a storage space of 50 MB and run the selection algorithm. Figure 10 shows the indexing recommendations after BJI selection. We find more information as the number of queries do not benefit from the fragmentation, indexed attributes, storage cost of selected BJI, the cost of queries before and after indexing, etc. In the same way as for the fragmentation, if DWA is satisfied with recommendations, he asks OptAssist to generate indexing scripts, otherwise he can go back for other choices.

6 CONCLUSION

Data warehouse physical design task has become a major issue. This is due to the characteristics of data warehouses: large volume, complexity of OLAP queries, the requirements of reasonable response time and management changes. In this environment, we have highlighted the difficulties that an administrator might encounter during optimization. These difficulties are numerous, because they involve multiple levels: choice of optimization techniques for all relevant queries to optimize, choice of selection mode, choice of algorithms and their parameters. Given these difficulties, we have identified the need to develop an advisor tool assisting DWA to make the right choices for optimization. We proposed *OptAssist* tool, offering three optimization techniques: primary horizontal partitioning, derived horizontal partitioning and bitmap join indexes. It can select these techniques in isolated or multiple modes and allows the DWA to select different algorithms and their parameters. Another particularity of *OptAssist* is that it offers a personalized and non-personalized optimization. It will be interesting to extend our tool by considering other optimization techniques, like materialized views, parallel processing, clustering, etc. Another possible extension is to consider other selection algorithms like ant colonies, taboo search, etc.

REFERENCES

[1] S. Agrawal. Automatic sql tuning in oracle 10g. In Proceedings of the 30th International Conference on Very Large Databases (VLDB), 2004.

[2] S. Agrawal. Database tuning advisor for microsoft sql server 2005. In Proceedings of the 30th International Conference on Very Large Databases (VLDB), 2004.

[3] K. Aouiche, J. Darmont, O. Boussaid, and F. Bentayeb. Automatic Selection of Bitmap Join Indexes in Data Warehouses. 7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK 05), August 2005.

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

[5] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias. A data mining approach for selecting bitmap join indices. Journal of Computing Science and Engineering, 2(1):206–223, January 2008.

[6] Ladjel Bellatreche, Kamel Boukhalfa, and Pascal Richard. Horizontal partitioning in data warehouse: Hardness study, selection algorithms and validation on oracle10g. in 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008), pages 87–96, September 2008.

[7] Ladjel Bellatreche, Rokia Missaoui, Hamid Necir, and Habiba Drias. A data mining approach for selecting bitmap join indices. Journal of Computing Science and Engineering, 2(1) :206–223, 2008.

[8] Kamel Boukhalfa. De la conception physique aux outils d'administration et de tuning des entrepˆots de donn´ees. Ph.d. thesis, Ecole Nationale Sup´erieure de M´ecanique et d'a´eronautique Poitiers et Universit´e de Poitiers, July 2009.

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

[10] C. Chee-Yong. Indexing techniques in decision support systems. Phd. thesis, University of Wisconsin - Madison, 1999.

[11] OLAP Council. Apb-1 olap benchmark, release ii. http ://www.olapcouncil.org/ research/ resrchly.htm, 1998.

[12] K. EL Gebaly and A. Aboulnaga. Robustness in automatic physical database design. in 11th International Conference on Extending Database Technology (EDBT'08), March, 2008.

[13] T. Johnson. Performance measurements of compressed bitmap indices. Proceedings of the International Conference on Very Large Databases, 1999.

[14] M. T. Özsu and P. Valduriez. Principles of Distributed Database Systems:Second Edition. Prentice Hall, 1999.

[15] A. Sanjay, V. R. Narasayya, and B. Yang. Integrating vertical and horizontal

[16] Partitioning into automated physical database design. Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 359–370, June 2004.

[17] A. Sanjay, C. Surajit, and V. R. Narasayya. Automated selection of materialized views and indexes in microsoft sql server. Proceedings of the International Conference on Very Large Databases, pages 496–505, September 2000.

[18] Zohreh Asgharzadeh Talebi, Rada Chirkova, Yahya Fathi, and Matthias Stallmann. Exact and inexact methods for selecting views and indexes for olap performance improvement. 11th International Conference on Extending Database Technology (EDBT'08), Mars 2008.

[19] D. C. Zilio, J. Rao, S. Lightstone, G. M Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. Db2 design advisor: Integrated automatic physical database design. Proceedings of the International Conference on Very Large Databases, pages 1087– 1097, August 2004.