The Workload You Have, the Workload You Would Like

Matteo Golfarelli
DEIS - University of Bologna
golfare@csr.unibo.it

Ettore Saltarelli
DEIS - University of Bologna
saltaett@csr.unibo.it

ABSTRACT

Real data warehouse workloads are often too large and complex to be directly optimized using the algorithms proposed in the literature for view materialization and indexing. In this paper we propose the idea of profile as an instrument for summarizing the workload features in order to help the designer to make the right choices. The ability of the profile to characterize a workload is then exploited to move backward using it as an input for an algorithm that generates a set of queries presenting the desired features. The algorithm proposed is finally used for creating the workloads necessary for testing the correspondence between different profiles and the results of optimization.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Information Storage and Retrieval - *Content Analysis and Indexing*; H.2.7 [**Information Systems**]: Database Management - *Database Administration*

General Terms

Design, Performance, Algorithms.

Keywords

Statistical indicators, Clustering, Logical Design, Optimization.

1. INTRODUCTION

During the design of a data warehouse (DW), the phases aimed at improving the system performance are mainly the logical and physical ones. A basic requirement for DW users is to obtain quick answers for their queries and the two main techniques to reach this goal are *view materialization* and *indexing*. Most of the approaches proposed in the literature rely on the existence of a reference workload that represents the target for the optimization [10]. Unfortunately, real workloads are much larger than those that can be handled by these techniques and thus view materialization and indexing remain tasks whose success depends on the experience of the designer that may find acceptable solutions by adopting rules of thumb and applying the trial-and-error approach.

The gap between academic approaches and real systems could be filled by techniques capable of determining on the one hand, the workload characteristics and maintaining, on the other, a reduced computational complexity.

Figure 1 shows the framework we assume for our approach: OLAP applications generate SQL queries whose logs are periodically elaborated for optimizing the system, but instead of directly using the workload extracted from the logs, a pre-optimization phase is carried out in order to determine: 1) a set of statistical indicators that profiles the workload and determines its main features, 2) a clustered workload that is representative of the original one and can be handled by view materialization and indexing algorithms.

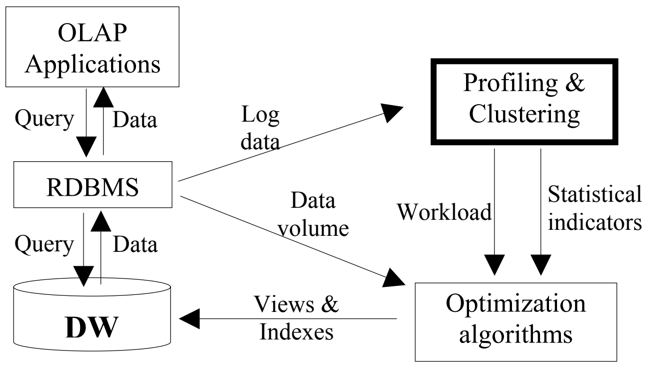

Figure 1: Overall framework for the optimization process

In particular, the optimization process can be driven by the profile that shows the best choices to the designer. For example, knowing if the workload is more suitable for being optimized by using view materialization or indexing will make it possible to assign more disk space to one of the two techniques.

In a previous work [4] we sketched, limitedly to the aspects related to view materialization, a set of statistical indicators that proved to be representative of the full workload and we defined a clustering algorithm capable of reducing the cardinality of the workload, maintaining at the same time the features of the original one.

This paper, based on the previous results, presents a complete study on the concept of workload profile, proposing in particular two original contributions: first, the workload profile is completed considering the statistical indicators related to indexes and discussing how the whole profile should be interpreted in order to understand at best the workload characteristics. Our qualitative considerations are tested against the TPC-H/R benchmark [11] evaluating the relationship between the profile and the execution cost of different workloads. Furthermore, we move backwards and propose a workload generation algorithm that produces a GPSJ [6] workload that respects a given profile. The generated workload can be used for testing and benchmarking purposes. It is remarkable that, unlike other fields like pattern recognition and operational research, this is the first attempt in the database field to define large and ad-hoc workloads for benchmarking (even well-known benchmarks like TPC-H/R are supplied with a limited and fixed set of queries). The ability to easily create large workloads with specific features can strongly reduce the time devoted to testing and can be very useful in evaluating the performances in different situations (e.g. fine or coarse aggregation, selective or unselective queries).

To the best of our knowledge no study, except the preliminary result proposed in [4], focused specifically on defining a profile for understanding the workload; nevertheless the idea of identifying common characteristics to improve the system performance have been largely used. In particular, clustering techniques have been proposed for defining groups of similar queries: in [3] the authors used clustering to reduce the complexity of the plan selection task. The concept of similarity is based on a complex set of features that it is necessary to encode when different queries can be efficiently solved using the same execution plan. This idea has been implicitly used in several previous works where a global optimization plan was obtained given a set of queries [9]. Another field where query similarity and clustering are often used is Information Retrieval since search engines need to cluster similar queries to identify FAQ and to better organize hierarchies of documents [12].

The rest of the paper is organized as follows: Section 2 presents the necessary background, Section 3 defines the statistical indicators for workload profiling; Section 4 presents the algorithm for generating the workload, while in Section 5 a set of experiments, aimed at proving the profile effectiveness, are reported. Finally, in Section 6 the conclusions are drawn.

2. BACKGROUND

It is recognized that DWs lean on the multidimensional model to represent data, meaning that indicators that *measure* a fact of interest are organized according to a set of *dimensions of analysis*; for example, sales can be measured by the `quantity sold` and the `price` of each sale of a given product that took place in a given store and on a given day. Each dimension is usually related to a set of attributes describing it at different aggregation levels; the attributes are organized in a *hierarchy* defined according to a set of functional dependencies. For example a product can be characterized by the attributes `PName`, `Type`, `Category` and `Brand` among which the following functional dependencies are defined: `PName`→`Type`, `Type`→`Category` and `PName`→`Brand`; on the other hand, stores can be described by their geographical and commercial location: `SName`→`City`, `City`→`Country`, `SName`→`CommmercialArea`, `CommercialArea`→`CommercialZone`.

In relational solutions, the multidimensional nature of data is implemented on the logical model by adopting the so-called *star scheme*, made up of a set of (fully denormalized) *dimension tables*, one for each dimension of analysis, and a *fact table* whose primary key is obtained by composing the foreign keys referencing the dimension tables. The most common class of queries used to extract information from a star schema are GPSJ [6] that consist of a selection, over a generalized projection over a selection over a join between the fact table and the dimension table involved. Within the scope of this paper we will assume that selections are always conjunctive range statements expressed on attributes in the dimension tables.

It is easy to understand that grouping heavily contributes to the global query cost and that such a cost can be reduced by precomputing (materializing) that aggregated information that is useful to answer a given workload. Unfortunately, in real applications, the size of such views never fits the constraint given by the available disk space and it is very hard to choose the best subset to be actually materialized. When working on a single fact scheme and assuming that all the measures contained in the elemental fact table are replicated a view is completely defined by its aggregation level.

Definition 1 The *pattern* of a view consists of a set of dimension table attributes such that no functional dependency exists between attributes in the same pattern.

Possible patterns for the sales fact are: P_1 = {`Month`, `Country`, `Category`}, P_2 = {`Year`, `Sname`}, P_3 = {`Brand`}. In the following we will use indifferently the terms pattern and view and we will refer to the query pattern as the coarsest pattern that can be used to answer the query.

Definition 2 Given two views V_i, V_j with patterns P_i, P_j respectively, we say that V_i can be *derived* from V_j ($V_i \leq V_j$) if the data in V_i can be calculated from the data in V_j.

Derivability determines a partial-order relationship between the views, and thus between patterns, of a fact scheme. Such partial-order can be represented by the so-called *multidimensional lattice* [1] whose nodes are the patterns and whose arcs show a direct derivability relationship between patterns. The multidimensional lattice is often used to efficiently store the information relevant to the materialization process but also provides an intuitive representation of the distribution of the queries with respect to their aggregation level.

Definition 3. We denote with $P_i \oplus P_j$ the least upper bound (*ancestor*) of two patterns in the multidimensional lattice.

Intuitively the ancestor of two patterns corresponds to the coarsest one from which both can be derived. Given a set of queries the ancestor operator can be used to determine the set of views that are potentially useful to reduce the workload cost (*candidate views*). The candidate set can be obtained, starting from the workload queries, by iteratively adding to the set the ancestors of each couple of patterns until a fixed point is reached.

As for indexing, the index types considered are B^+-tree and bitmap; we will assume that a B^+-tree index is always built on the key of each table present in the DW, additional indexes can be built on single attributes belonging to the key of a fact table (to speed up join) and on attributes in the dimension tables (to speed up selections).

The cost model we adopted for evaluating the workload cost is the one proposed in [8]: the execution plan is obtained by making up a set of base operators corresponding to the base operations (e.g. access a table, access to an index, execute a join, etc.) whose cost is expressed in terms of disk pages accessed to retrieve the data. The set of rules used by the authors to determine the best plan has been extended by relaxing the constraint of using an existing index, regardless of the negative impact it could have on the query cost. Evaluating, each time, the usefulness of an index makes the optimizer model more realistic.

3. PROFILING THE WORKLOAD

Profiling means determining the values of a set of statistical indicators that captures the workload features that have an impact on the effectiveness of different optimization techniques. In particular, we are interested in those relevant to the problem of view materialization and indexing that help the designer to answer queries like:

- *"How suitable is the workload for materialization/indexing ?"*
- *"How difficult will it be to find the best set of views to be materialized ?"*
- *"In percentage, how much disk space should be allocated to indexes or views?"*

Given a set of queries $\mathcal{W}=\{Q_1,\ldots,Q_n\}$, a profile $R_\mathcal{W}$ is a quintuple (*AAL*, *ASK*, *AS*, *LSC*, *ACT*), the first two indicators analyze the behavior of the workload $\mathcal{W}$ with respect to aggregation, while the last three summarize its characteristics in terms of selection criteria. In the rest of the paper we will refer to $R_\mathcal{W}[i]$ as the *i*-th indicator in the profile. We now present each indicator separately discussing how its values should be interpreted individually and with respect to the other ones.

3.1 Indicators on Aggregation Level

AAL and *ASK* are based on the concept of cardinality of the views associated to a given pattern that can be estimated knowing the data volume of the fact scheme that we assume to contain the cardinality of the base fact table and the number of distinct values of each attribute in the dimension tables.

The cardinality of an aggregate view can be estimated using Cardenas' formula [2] which states that, when throwing N distinct objects into B buckets, the expected number of buckets in which at least one object will fall can be estimated as:

$$\Phi(B,N) = B\cdot\left(1-\left(1-\frac{1}{B}\right)^N\right) \le \min(B,N)$$

In our case the objects are the tuples in the elemental fact table with pattern P_0 (whose number $|P_0|$ is assumed to be known) while the number of buckets is the maximum number of tuples, $|P|_{Max}$, that can be stored in a view with pattern P and that can be easily calculated given the cardinalities of the attributes belonging to the pattern, thus

$$Card(P) = \Phi(|P|_{Max}, |P_0|)$$

The aggregation level of a pattern P is calculated as:

$$Agg(P) = 1 - \frac{Card(P)}{|P_0|}$$

$Agg(P)$ ranges between [0,1[, the higher the values the coarser the pattern. The average aggregation level of the full workload $\mathcal{W}=\{Q_1,\ldots Q_n\}$ can be calculated as

$$AAL = \frac{1}{n}\sum_{i=1}^{n} Agg(P_i)$$

where P_i is the pattern of query Q_i. *AAL* characterizes to what extent the information required by the users is aggregated and expresses the willingness of the workload to be optimized using materialized views. Intuitively, workloads with high values of *AAL* will be efficiently optimized using materialized views since they determine a strong reduction of the number of tuples to be read. Furthermore, the limited size of such tables allows a higher number of views to be materialized.

Measuring the aggregation level is not sufficient to characterize the workload; in fact workloads with similar values of *AAL* can behave differently, with respect to materialization, depending on the attributes involved in the queries. Consider for example two workloads $\mathcal{W}_1=\{Q_1, Q_2\}$ and $\mathcal{W}_2=\{Q_3, Q_4\}$ formulated on the sales fact scheme and the pattern of their queries:

• P_1 = {`Category, City`}	Card(P_1) = 2100
• P_2 = {`Type, Country`}	*Card*(P_2) = 1450
• P_3 = {`Category, Country`}	*Card*(P_3) = 380
• P_4 = {`Brand, CommercialZone`}	*Card*(P_4) = 680

Materializing a single view to answer both the queries in the workload is much more useful for $\mathcal{W}_1$, than for $\mathcal{W}_2$ since in the first case the ancestor is very "close" to the queries ($P_1 \oplus P_2$={`Type, City`}) and still coarse, while in the second case it is "far" and fine ($P_3 \oplus P_4$={`SName, PName`}). This difference is captured by the distance between the two patterns that we calculate as:

$$Dist(P_i, P_j) = Agg(P_i) + Agg(P_j) - 2\,Agg(P_i \oplus P_j)$$

$Dist(P_i, P_j)$ is calculated in terms of distance of P_i and P_j from their ancestor that is the point of the multidimensional lattice closest to both the views. Figure 2 shows two different situations on the same multidimensional lattice: even if the aggregation level of the patterns is similar, the distance between each couple changes considerably.

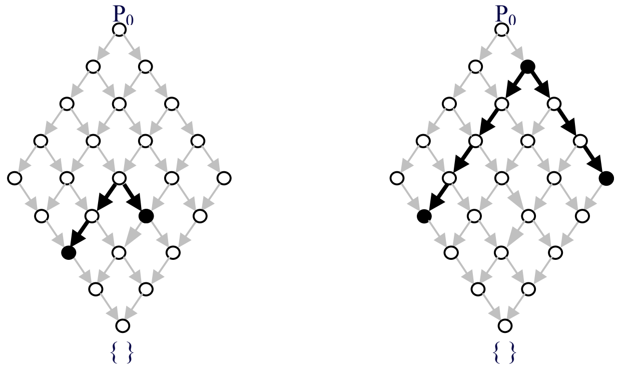

Figure 2: Distance between close and far patterns

The average skewness of the full workload $\mathcal{W}=\{Q_1,\ldots Q_n\}$ can be calculated as

$$ASK = \frac{2}{n\cdot(n-1)}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Dist(P_i, P_j)$$

where P_z is the pattern of query Q_z; *ASK* ranges in [0,2[[1] . Intuitively, workloads with low values for *ASK* will be efficiently optimized using materialized views since the similarity of the query patterns makes it possible to materialize few views to optimize several queries.

[1] The maximum value for *ASK* depends on the cardinalities of the attributes and on the functional dependencies defined on the hierarchies, thus it cannot be defined without considering the specific star schema.

3.2 Indicators for Selectivity

In order to characterize a workload in terms of its selectivity we propose three indicators that have proved to be representatives of the factors that influence the execution cost. The analysis of the selectivity indicators is harder since the evaluation of their values must be based on the values of those of the aggregation ones.

The main indicator is the average selectivity that is calculated as

$$AS = \frac{1}{n}\sum_{i=1}^{n} sel(Q_i)$$

where the selectivity of each query is defined as $sel(Q_i) = \prod_{j=1}^{m_i} sel(Q_{i,j})$ where m_i is the number of conditions for Q_i and $sel(Q_{i,j})$ is the selectivity of the j-th statement (i.e. percentage of the attribute values selected by the statement); *AS* ranges in [0,1]. Intuitively, workloads with lower values of *AS* will require a stronger use of indexes; on the other hand, selectivity itself is not sufficient to characterize the impact of indexing with respect to materialization since it depends on "where" and "how" the statements are formulated. In fact, if the statements are formulated on strongly aggregated data the effect of materialization could be stronger, regardless of the level of selectivity. To emphasize this effect a second operator to be considered is the coefficient of the least-square line (*LSC*) defined on the *aggregation-selectivity plan* by the set of points, one for each query Q_i, having as x-values $Agg(Q_i)$ and as y-values $Sel(Q_i)$ (see Figure 3). When *LSC* is near to zero selectivity is equally distributed at the different aggregation levels, while when *LSC* is positive/negative selectivity is stronger for queries with lower/higher aggregation level. Compatibly with the values of *AS*, workloads with positive values of *LSC* will be more inclined to be optimized using indexes with respect to those with negative values since the effect of selections concerns tables with higher cardinalities.

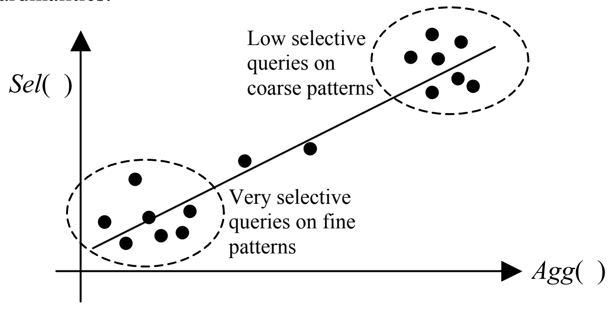

Figure 3: The mean-square error line for the aggregation-selectivity plan

Finally, a third criteria affecting the usefulness of indexes is the average number of constrained tables:

$$ACT = \frac{1}{n}\sum_{i=1}^{n} m_i$$

In fact, given a selectivity value *AS*, workloads with higher values of *ACT* will require a higher number of indexes to apply all the conditions profitably. It should be noted that, even if all the indexes required are available, indexing will be less effective with respect to the case with low values for *ACT*. In fact, for a fixed *AS*, a high value of *ACT* implies unselective conditions that require more data to be read.

4. GENERATING THE WORKLOAD

Generating a GPSJ workload $\mathcal{W}$ over a fact scheme F means choosing the set of feasible queries over F such that $R_{\mathcal{W}}$ does not differ more than ΔR from a target profile R_{opt} given as input together with the desired cardinality $|\mathcal{W}|$. As shown in Figure 4, generation is a two-step process: initially the set of aggregation patterns $\mathcal{P}$ respecting the target profile are searched in the multidimensional lattice space, then selectivity criteria are applied. It should be noted that the workload definition process does not consider measures assuming that they are always requested together; this assumption is justified since optimizations based on vertical fragmentation of fact tables are rarely considered.

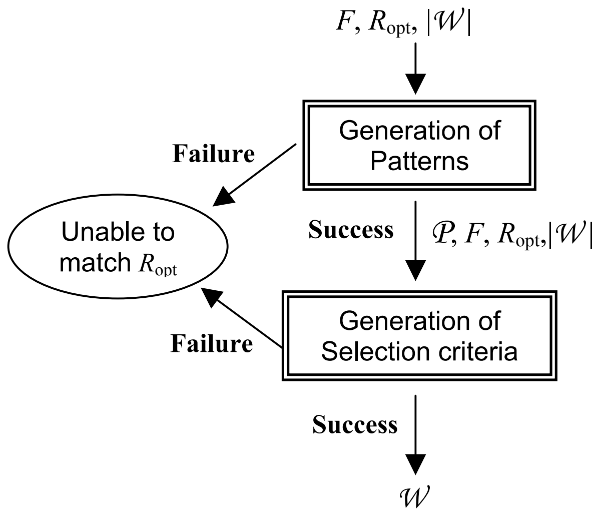

Figure 4: The workload generation process

4.1 Generation of Patterns

$\mathcal{P}$ is determined on the multidimensional lattice induced by the fact hierarchies using a taboo-search approach that, at each iteration, adds a query to the solution by descending the multidimensional lattice, starting from the root, with a depth-first rule until a pattern respecting R_{opt} is found. In order to reduce the dimension of the search space only the most promising child of each node is considered (i.e. the child that produce the most significant improvement in the solution profile). When no feasible pattern is found the partial solution is modified by removing the worst pattern (i.e. the pattern that determines the maximum increase for ΔR) that is then added to a taboo-list. A partial solution R respects R_{opt} if

$$| R[i] - \Delta R[i] | < TS[i] \qquad i = 1,..2$$

where $TS[i]$ are thresholds defined by the designer.

The search fails when no new pattern can be added and the partial solution cannot be modified. Failures can be due to the heuristic nature of the approach necessary to limit the dimension of the search space that is exponential in the number of attributes in the hierarchies but, more frequently, fails since the characteristics of the fact do not match the target profile. In other words, it is not at all possible to find $|\mathcal{W}|$ queries with profiles R_{opt}.

4.2 Generation of Selection Criteria

Given the set of aggregation pattern $\mathcal{P}$ for the queries in the workload the selectivity $sel(Q_i)$ of each query can be set on the aggregation-selectivity plan by projecting on the ordinate axes the values of $Agg(Q_i)$ by means of the line passing through (AS, AAL) and having LSC as an angular coefficient (see Figure 5). In fact, it is possible to prove that the points of the line having x-coordinate $Agg(Q_i)$ will have as y-coordinate the selectivity values that determine the correct AS in the target profile.

The single selectivity statements are then defined according to the values of ACT and compatibly with the cardinalities of attributes in the aggregation patterns. In fact, since it is not always possible to assign the theoretical selectivity (e.g. a selectivity of 10% on an attribute with 4 distinct values) a compensation algorithm is required. Compensation increases/decreases the selectivity of the other statements in order to bring AS back to the input value. The algorithm is first applied on the other selections of the same query and then on the statements of the other ones.

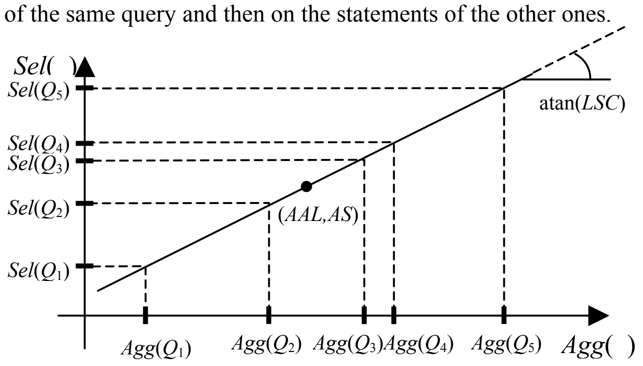

Figure 5: The line used to infer the selectivity of the queries given their aggregation levels.

5. TESTS AND DISCUSSION

In this section we propose a set of tests aimed at proving the effectiveness of profiling and we discuss how the indicators should be interpreted as a whole, that is, considering also how the meaning of one of them is influenced by the values of the others. Test have been carried out on the TPC-H/R benchmark [11] and in particular on the LINEITEM fact scheme; as already said the cost function is the one proposed in [8] that quantifies the workload cost in the number of disk pages accessed to answer the query. The fact schema data requires 1.1 Gb considering also the space used for indexing the primary keys of the tables.

Effectiveness of profiling is measured indirectly analyzing how workloads with different profiles respond to optimizations like view materialization and indexing. View materialization is carried out using the classic approach proposed by Baralis et al.[1]: the algorithm first determines the set of candidate views and then heuristically chooses the best subset that fits given space constraints. Splitting the process into two phases allows us to estimate both the difficulty of the problem, that we measure in terms of the number of candidate views, and the effectiveness of materialization that is calculated in terms of the number of disk pages saved by materialization. As for indexing we used the heuristic approach proposed in [5] that, given a set of materialized views, determines the set of useful indexes (and their types) first, then chooses the optimal subset based on the saving per disk page determined by each index.

5.1 Profiling Small Workloads

This set of tests has been carried out on workloads containing 20 queries. The limited cardinality allows the optimization algorithms to be applied without pre-reducing the number of queries to be considered.

Table 1 shows workloads characterized by no selections and different profiles for the indicators relative to materialization. It is evident that the number of candidate views, and thus the complexity of view materialization, mainly depends on the values of ASK and is slightly more influenced by AAL. The simplest workloads to be elaborated will be those with highly aggregated queries with similar patterns, while the most complex will be those with very different patterns with a low aggregation level. The effect of a high skewness is more evident as soon as the cardinality of the workloads is increased: those with a "nice" profile still perform well, while the others quickly become too complex. Figure 6 shows how the workloads perform with respect to materialization: Figure 6.a indicates that, regardless of the difficulty of the problems, workloads with high values of AAL are strongly optimized even when a limited disk space is available for storing materialized views. This behavior is induced by the dimension, and thus by the number, of the materialized views that fits the space constraint as can be verified in Figure 6.b.

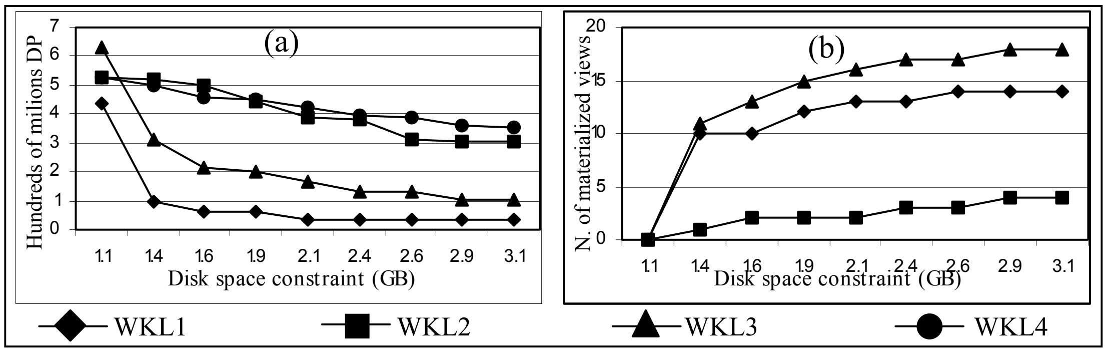

Figure 6: Cost of the workloads (a) and number of materialized views (b) on varying the disk space constraint.

Table 1. Profiles of different 20/30-queries workloads with no selections. Unchanged parameters in gray.

Name	N. of Queries	*AAL*	*ASK*	*AS*	*LSC*	*ACT*	N. Cand. views	N. Mat. views (at 2GB)
WKL1	20	0.835	0.348	0	0	0	97	15
WKL2	20	0.186	0.327	0	0	0	124	2
WKL3	20	0.790	0.810	0	0	0	596	15
WKL4	20	0.384	0.751	0	0	0	868	2
WKL5	30	0.884	0.316	0	0	0	99	14
WKL6	30	0.352	0.668	0	0	0	>36158	uncompleted after 2 days

Table 2. Profiles of different 20-query workloads completed with selections. Unchanged parameters in gray.

Name	*AAL*	*ASK*	*AS*	*LSC*	*ACT*	View save	Index save	V-I space trade-off at 2.1 GB
WKL1a	0.835	0.348	0.04	0	2	96.7 %	2.1 %	59% - 41%
WKL1b	0.835	0.348	0.25	0	2	88.2 %	7.8 %	84% - 16%
WKL1c	0.835	0.348	0.5	0	2	88.9 %	4.9 %	84% - 16%
WKL4a	0.384	0.751	0.04	0	2	27.3 %	52.6 %	77% - 23%
WKL4b	0.384	0.751	0.25	0	2	28.1 %	48.0 %	68% - 32%
WKL4c	0.384	0.751	0.5	0	2	22.0 %	29.9 %	67% - 33%

Table 3. View & index save for workloads with different values of *LSC* and *ACT*. Unchanged parameters in gray.

Name	*AAL*	*ASK*	*AS*	*LSC*	*ACT*	View save	Index save
WKL7a	0.542	0.607	0.349	0.8	1.0	61.8 %	14.7 %
WKL7b	0.542	0.607	0.366	-0.8	1.1	54.3 %	0.26 %
WKL7c	0.542	0.607	0.3	0.0	1.2	25.9 %	62.9 %
WKL7d	0.542	0.607	0.29	0.1	2.8	18.0 %	62.2 %

Let us now consider the indicators concerning selectivity and in particular *AS*. Table 2 reports the profiles of the previous workloads extended with select conditions: while *LSC* and *ACT* remain constant, *AS* is varied ranging from 0.04 (very selective workload) to 0.50 (not very selective workload);

The effects on the optimization effectiveness are measured calculating in which percentage the initial workload cost is reduced due to the effects of materialization (view save) and indexing (index save). The values are obtained as the average of different tests in which the space constraint is varied from 1.1 to 3.1 Gb. For each test the values considered are those relative to the optimal trade-off between the space used for materialization and indexing.

It is evident that in general materialization is more effective than indexing that becomes relevant only when *AS* is very high and the characteristics of the workload make it unsuitable for materialization. Consider for example WKL1a and WKL4a: even if they are both very selective only the second one greatly benefits from indexing since for the first one the cost is already strongly reduced by materialization. It should be noted that according to the parameters, the percentage of the disk space that should be devoted to indexing and materialization also varies. This information is very important for the designers since materialization and indexing are carried out in two separate steps while the splitting up of the space must be done a priori: assigning the wrong quantity of space to one of the two phases may induce suboptimal performances and a waste of a very limited resource. The last column in Table 2 shows how the percentages change when 2Gb of disk space is available for the DW; the best trade-off depends on both aggregation and selectivity features; in particular, more space should be used for indexing when the profile shows that the workload is not suitable for being optimized by optimization and when selectivity is higher.

Finally, in Table 3 *LSC* and *ACT* are analyzed using a set of workloads sharing the same characteristics in terms of aggregation level, skewness and selectivity; while the *AAL* and *ASK* have average values, *AS* is low in order to show up the results of the analysis. As for *LSC*, the best optimization takes place when the queries insisting on the finest views are strongly selective (*LSC* positive) since they will be successfully optimized by indexing while views will be used for unselective, but aggregate, queries. On the other hand, workloads with few but very selective conditions (low values of *ACT*) are well optimized using indexes since few index and data pages must be read to apply the conditions.

5.2 Profiling Large Workloads

Tests reported in this section are aimed at evaluating how clustering impacts on the relationship between profiling and optimization. Given that large workloads must be pre-reduced in order to apply optimization algorithms, it is important to verify to what extent such transformation impacts on their characteristics and thus how effective optimization will be. In our test we adopted the clustering algorithm proposed in [4] that is based on a hierarchical approach that recursively agglomerates the two most similar clusters that contain one single query at the beginning. The representative of a cluster is a query whose pattern is the ancestor

Table 4. Effects of clustering on the optimization process

Name	N. of Clusters	# Cand. Views	Total save	View save	Index save	V-I space trade-off at 2.1 GB
	20	2125	85.8 %	35.2 %	50.6 %	55 % - 45 %
WKL8	15	499	85.5 %	33.9 %	51.6 %	55 % - 45 %
	10	129	80.7 %	5.1 %	75.6 %	54 % - 46 %
	20	4744	87.4 %	85.9 %	1.8 %	71 % - 29 %
WKL9	15	2136	86.9 %	85.0 %	1.9 %	77 % - 23 %
	10	384	84.6 %	81.6 %	3.0 %	61 % - 39 %

of the queries in the cluster and whose select condition is the logical conjunction of the statements of the queries belonging to the cluster. Table 4 reports the optimization results for two workloads: WKL8 (*AAL*=0.378, *ASK*=0.738, *AS*=0.10,LSC=0.0, *ACT*=1.8) and WKL9 (*AAL*=0.915, *ASK*=0.209, *AS*=0.75, *LSC*=0.0, *ACT*=1.0) both containing 200 queries. As concerns materialization, profile of WKL9 predicts that it is more willing to be optimized using materialized views (i.e. *AAL* high and *ASK* low), accordingly the view save is much stronger than that of WKL8. On the other hand, WKL8 benefits much more by indexing as we could infer from the selectivity indicators. Accordingly to the previous results the percentage of space devoted to view materialization is definitely higher for WKL9 than for WKL8. Finally, while the complexity of the problems strongly depends on the number of clusters, the total optimization reduction is slightly affected thus proving that the clustering algorithm preserves original workload features.

Finally, while the complexity of the problems strongly depends on the number of clusters, the total optimization reduction is slightly affected thus proving that the clustering algorithm preserves original workload features. Of course, the distortions induced in the workload features become more evident when the number of clusters is strongly reduced. For example Table 2 shows that, as concerns materialization, WKL8 cannot be effectively represented by 10 clusters; fortunately 10 clusters are still enough to successfully carry out indexing.

6. CONCLUSIONS

In this paper we proposed the idea of profiling a DW workload in order to summarize its characteristics. The profile can be used on the one hand to help the designer during logical and physical optimization since we have proved that it captures many relevant features. On the other hand, we proposed an algorithm for generating workloads starting from a given profile. This ability can strongly reduce the time devoted to testing and can be very useful in evaluating the performances in different situations. Our future work will consist in making the qualitative information obtained from the profile more “quantitative”; in particular we will define a set of curves, parametric with respect to the statistical indicators, to estimate how the workload will respond to optimization as a function of variables like available disk space and response time desired. Producing as output a curve will strongly simplify the interpretation of the indicators that is made difficult by their intrinsic interdependence: the curve will condense in a unique and graphic representation all the information carried by the set of indicators.

REFERENCES

[1] E. Baralis, S. Paraboschi and E. Teniente. Materialized view selection in a multidimensional database. In Proc. 23rd VLDB, Greece, 1997.

[2] A.F. Cardenas. Analysis and Performance of Inverted Database Structures. Communications of the ACM, 18(5):253–263, 1975.

[3] A. Ghosh, J. Parikh, V.S. Sengar and J. R. Haritsa. Plan Selection Based on Query Clustering, In Proc. 28th VLDB, Hong Kong, China, 2002.

[4] M. Golfarelli. Handling large workloads by profiling and clustering. To appear in Proc. 5th DaWaK 2003, Prague, 2003.

[5] M. Golfarelli, S. Rizzi and E. Saltarelli. Index selection for data warehousing. In Proc. DMDW'2002, Toronto, Canada, pp. 33-42, 2002.

[6] A. Gupta, V. Harinarayan and D. Quass. Aggregate-query processing in data-warehousing environments. In Proc. 21st VLDB, Switzerland, 1995.

[7] T. P. Nadeau and T. J. Teorey. Achieving scalability in OLAP materialized view selection. In Proc. DOLAP’02, Virginia USA, 2002.

[8] S. Rizzi and E. Saltarelli. View materialization vs. Indexing: balancing space constraints in Data Warehouse Design. To appear in Proc. CAISE’03, Austria, 2003.

[9] T. K. Sellis. Global query Optimization. In Proc. SIGMOD Conference Washington D.C. 1986, pp. 191-205

[10] D. Theodoratos, M. Bouzeghoub. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In Proc. DOLAP’00, Washington D.C. USA, 2000.

[11] Transaction Processing Performance Council. TPC Benchmark H (Decision Support) Standard Specification, Revision 1.1.0, 1998, http://www.tpc.org.

[12] J-R Wen, J-Y Nie and H-J Zhang. Query clustering using user logs. ACM TOIS, Vol. 20, N. 1, Jan 2002, pp. 59-81.