A Quality-Based Framework for Physical Data Warehouse Design

Mokrane Bouzeghoub, Zoubida Kedad
Laboratoire PRiSM, Université de Versailles
45, avenue des Etats-Unis
78035 Versailles Cedex, France
Mokrane.Bouzeghoub@prism.uvsq.fr, Zoubida.Kedad@prism.uvsq.fr

Abstract

Data warehousing is a software infrastructure which supports OLAP applications by providing a collection of tools which allow data extraction and cleaning, data integration and aggregation, and data organization into multidimensional structures which are suitable for decision making. At the design level, a data warehouse is defined as a hierarchy of view expressions. One key problem of the design of a data warehouse is the selection of a set of views to materialize in order to meet the performance requirements, that is, to minimize both the evaluation cost of the users queries and the maintenance cost of the materialized expressions. Existing approaches either propose algorithms that explore the search space by enumerating and evaluating all the possible solutions, or introduce heuristics that restrict the search space. In this paper, we propose a framework to support the physical design of a data warehouse; it is based on a logical model, called the view synthesis graph. A level of overlapping is defined to characterize the graph. This level allows to order the search space. The framework integrates some quality factors that can be used either to drive the exploration of the search space, or to reduce the search space, or both.

1. Introduction

Data warehousing is a software infrastructure which supports OLAP applications by providing a collection of tools which (i) collect data from a set of distributed heterogeneous sources, (ii) clean and integrate this data into a uniform representation, (iii) aggregate and organize this data into multidimensional structures which are suitable for decision making, and (iv) refresh it periodically to maintain the data up to date and accurate. Many problems related with this approach have been addressed, such as data extraction and cleaning, selection of the best views to materialize, update propagation, and multidimensional representation and manipulation of data.

View materialization is a physical design technique which has been introduced to optimize database queries by making persistent some intermediate results, possibly shared by different queries [Gupta & Mumick 95]. It is seen as a caching technique which contributes to optimize queries. Additionally, in the data warehousing context, view materialization avoids overloading sources that supply operational data, and therefore makes data warehouse applications partially and momentarily independent of the data sources. This introduces two complementary constraints: (i) the first requires that all OLAP queries must be evaluated using only the materialized views, (ii) the second requires that materialized views must be updated periodically with respect to data changes OLAP applications want to see, and to a level of freshness required for this data. Consequently, one key issue during the physical design of a data warehouse is the selection of the set of views to materialize in order to optimize both the query evaluation and the maintenance cost.

The general approach to select the best views to materialize in a data warehouse consists in exploring a solution space which is defined by all the plans derived from a multi-query graph which integrates all the input queries. Two kinds of exploring techniques are reported in the literature: the first is an exhaustive evaluation of all the possible materialization scenarios [Theodoratos & Sellis 97], the second proposes heuristics to reduce the size of the search space [Theodoratos et al 99][Yang et al 97][Ligoudistianos et al 99]. The algorithms of the second class are obviously less expensive compared to those of the first class, however, they do not guarantee to derive the best solution as the search space is arbitrarily reduced using some heuristics.

Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000)
Stockholm, Sweden, June 5-6, 2000
(M. Jeusfeld, H. Shu, M. Staudt, G. Vossen, eds.)
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/

Algorithms of both classes are driven by cost functions that formalize the compromise between the query evaluation cost and the maintenance cost, called the operational cost. Our general comments on these approaches are the following:

- Both approaches search for the materialization scenario having the minimal operational cost, without considering whether this minimal cost coincides or not with user needs. In practical applications, it is often not necessary to search for the best solution but for the solution that best fits the user requirements. Indeed, it is not necessary to spend too much time to find the solution with a minimal cost while an intermediate solution whose cost fits user requirement may be sufficient. On the other side, if the solution having the minimal cost is far from user needs, it's important to highlight the mismatch between what the user requires and what the technology is able to provide, especially when the costs of all the existing solutions are monotonic and predictable.

- Heuristic-based approaches provide faster algorithms to derive reasonable solutions but do not provide any characterization of these solutions with respect to those derived from the exhaustive search. Besides this point, they do not provide any other quality factor - e.g. flexibility or evolution - which can convince that the derived solution is not useless.

- It is hard in both approaches to introduce additional requirements in the corresponding algorithms, such as concurrency control and security for example, which may also influence the operational cost.

A very few research has been done in handling quality factors in logical and physical data warehouse design. A substantial effort has been done within the European DWQ project where a generic model of quality has been defined [Jarke et al 99] [Vassiliadis et al 99] [Jeusfeld et al 98] to capture the set of quality factors associated to the various data warehouse objects, and to perform their measurement in a specific quality goal. It was particularly shown that the refreshment process [Bouzeghoub et al 99] and the selection of the materialized views [Bouzeghoub & Theodoratos 99] are demonstrative examples where quality can drive the design process. Following the work presented in [Bouzeghoub & Theodoratos 99], this paper is a step further to elaborate a general framework which allows to integrate in a more systematic way several quality factors into the physical design process.

This paper is organized as follows: in section 2, we present the general framework for physical design. Section 3 gives some preliminary definitions of the fundamental concepts that constitute the baseline of our framework. Section 4 presents the use of this framework for the selection of the best materialization scenario. Section 5 presents the related works and section 6 concludes with further research.

2. The General Framework for Quality-Based Physical Design in Data Warehousing

As recalled before, algorithms for materialized view selection are generally based on a cost function that combines query evaluation cost and view maintenance cost. Given a cost function, the selection problem is posed as an exhaustive search of the best materialization scenario, possibly driven by some heuristics that restrict the search space under consideration. The search space itself is generally represented as a multi-query graph, which is the union of all possible plans of all user views (or queries).

The idea behind our framework is to make the materialized view selection problem as practical as possible by providing a context within which derived solutions can be evaluated with respect to their relevance to user requirements in terms of quality factors. Quality factors are considered as a good way to handle non-functional requirements. They can play a central role in the algorithms which select a materialization scenario. Indeed, the complexity of the view selection problem is such that running the corresponding algorithms for real applications remains a challenge. Thus, introducing quality factors may, on the one hand, help in reducing the search space, and on the other hand, allow to characterize the derived solutions.

We assume that users are able to specify these quality factors in a formal way as numerical values, logical assertions or any other mathematical formula. As an example of requirement for the selection of materialized views, one may give boundaries to the operational cost, and the exploration procedure will therefore consider only solutions which match this requirement. Two categories of quality factors can be distinguished:

- Those who define or constrain the cost function, such as conformance to space limitation, upper limit to query response time, freshness of data;
- Those who impose some desired features on the derived materialization scenario (the selected plan), such as evolution, maintainability or traceability.

Although the second category of quality factors is not directly involved in the cost function, it may help in the generation of the heuristics which can drive a selective search, i.e., it might be a part of the selection algorithm.

The second category of quality factors constrains the structure of the views. They are generally related to the level of overlapping between view definitions. This overlapping level is very important in physical design; the higher it is, the more complex is the selection problem. Indeed, considering a set of n user views (or queries) whose plans do not overlap, the complexity of the selection problem is reduced to the complexity of the optimization of a list of n independent queries. Except for this very specific case, user views in data warehouse systems generally overlap. We assume that *the complexity of the materialized views selection algorithms is not independent of the level of overlapping* since any materialization scenario has a direct influence on the evaluation of all user views which share the same materialized views. Moreover, evolution of a physical design as well as traceability and many other different quality factors can also be correlated to the level of overlapping of views. A part of our design framework is consequently based on these structural features.

Figure 1 shows a simplified view of the framework, which uses a class of quality factors which may be defined on a given search space. This class can be divided in two subclasses, the first one concerning the quality factors which constrain the cost function, and the second concerning those which characterize the structural aspect of a materialization scenario. The quality factors of the latter subclass may be used to generate heuristics which restrict the search space or which allow to qualify the selected solutions. They can also be used to select a representation which satisfies some level of overlapping. This representation will then be used as a restricted search space on which the selection procedure is performed. Section 4 presents the usage of this framework through an example.

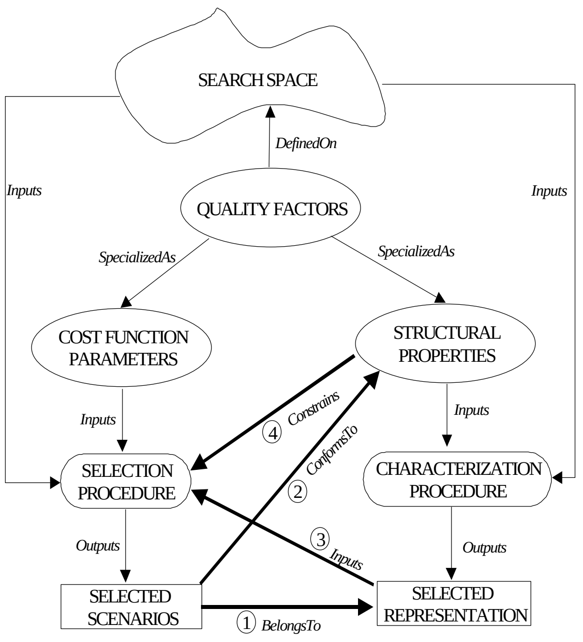

Figure 1: The general framework for quality based design

3. Preliminary definitions

In this section, we present the formalisms and the definitions used in the proposed framework for data warehouse design. We assume that the input queries are represented by relational expressions. In the remaining of the paper, these queries are called users views. Our framework is based on a graph representing the logical perception of a data warehouse, and called the view synthesis graph.

The design process of the logical schema of a data warehouse is out of the scope of this paper; it starts from the representation of the different user views. These views represent the users needs, that is, the aggregated data corresponding to the user requirements, computed using a given set of relations located in the data sources. We assume that the user views are represented by relational expressions. In a data warehouse, the user views are not necessarily disjoint. A given user view is represented by a set of equivalent relational expressions corresponding to different plans.

The result of this design process is the logical schema, called the view synthesis graph (VSG). The design process of the VSG produces successively the following representations:

- The multi-query graph (MQGs): we assume that each user view is a relational expression. Given such an expression, a set of equivalent plans is considered; they are represented by a MQG, which is presented in section 3.1.
- The multi-view graph (MVG): this graph represents the integration of all the MQGs corresponding to the set of user views. Therefore, it represents for each user view a set of relational expressions. The MVG is presented in section 3.2. The properties of this graph are given in section 3.3
- The view synthesis graph (VSG): this graph is generated from the MVG and it is such that each user view is represented by a single expression. This graph is presented in section 3.4, along with the definition of the level of overlapping.

One important feature of the VSG is that it shows a set of sub-expressions which are shared by several user views. The following sub-sections describe in more details the different graphs and their properties.

3.1. The multi-query graph

Let us consider a user view V_i. This view can be represented by a set of equivalent algebraic expressions denoted E_{Vi} and such that:

$$E_{Vi} = \{e_{1Vi}, e_{2Vi}, ..., e_{kVi}\}$$

The algebraic expressions corresponding to the user view V_i are denoted e_{kVi}. They use a set of operations

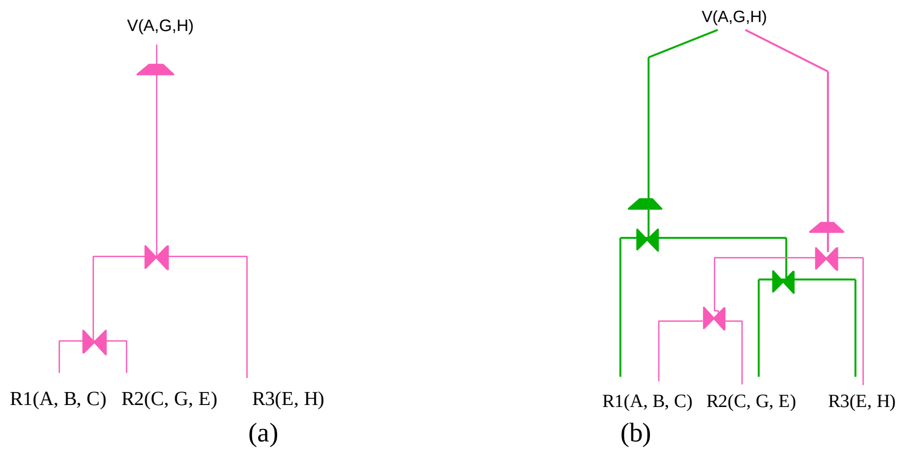

Figure 2 : The multi-query graph associated with a user view

such as join, project and restrict operations, or some aggregate function. Each expression represents a given plan for the user view V_i. In order to limit the number of considered plans, we will consider only those for which the project operation is the last one.

Given a user view V_i and the associated set of equivalent expressions E_{Vi}, the multi-query graph represents each element e_{jVi} of the set E_{Vi}. Each expression e_{jVi} is either an expression without any project operation, or an expression in which the project operation is the last one. Note that this restriction will reduce the number of expressions in the MQG, but not the number of common sub-expressions. The multi-query graph does not represent all the equivalent expressions to the initial view, because the search of these expressions may lead to an infinite set of expressions. This graph represents only a sub-set of the possible equivalent expressions which will be used as an input to generate the multi-view graph presented in the next section. The plans of the set E_{Vi} can be generated using the properties of the algebraic operations, such as distributivity or associativity. Figure 2 gives an example of a user view V and the associated multi-query graph.

3.2. The multi-view graph

Each user view being represented by a multi-query graph, the multi-view graph (MVG) consists in integrating all the MQGs into a single graph. It therefore represents a set of equivalent expressions for each user view. More formally, let us consider the following notations:
$V = \{V_1, V_2, ..., V_n\}$ the set of user views,
$E_{Vi} = \{e_{1Vi}, e_{2Vi}, ..., e_{kVi}\}$ the set of equivalent expressions corresponding to the view V_i.
The multi-view graph denoted MVG is such that:
$MVG = \{E_{V1}, E_{V2}, ..., E_{Vn}\}$

The multi-view graph is a set whose elements are sets of equivalent expressions associated with the user views.

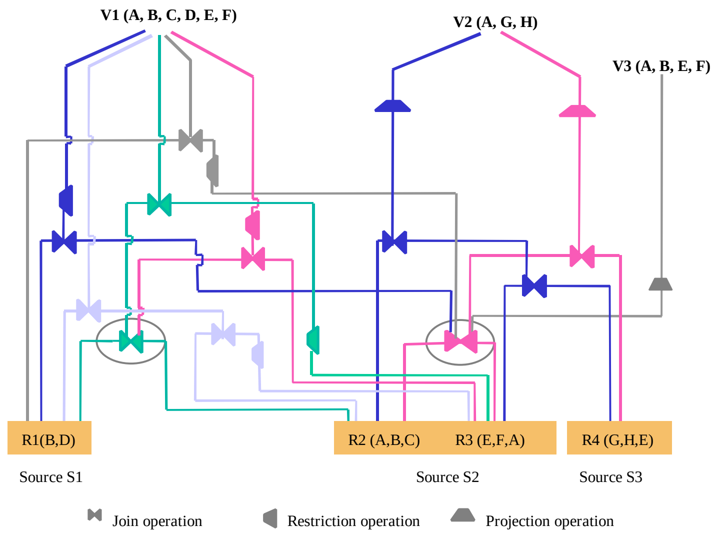

Figure 3 : Example of a multi-view graph (MVG)

Figure 3 shows an example of the graphical representation of an MVG, built by integrating the MQGs corresponding to three user views V_1(A, B, C, D, E, F), V_2(A, G, H) and V_3(A, B, E, F). In this example, we can notice that some sub-expressions are used by more than one user view. An ellipse represents these expressions.

The MVG graph represents a set of equivalent expressions for all the user views of the data warehouse. If we consider the graphical representation of a MVG, each node in the graph represents either a source relation, a sub-expression, or an expression corresponding to a user view. The graph shows all the existing sub-expressions used by each user view. Each sub-expression may be either shared by several plans, or used in only one plan.

3.3. Properties of the multi-view graph

The properties which characterize the sub-expressions and the plans in a MVG are defined below.

3.3.1. Level of sharing of a sub-expression

The level of sharing of a sub-expression is the number of distinct views which uses this sub-expression; let us consider the multi-view graph MVG corresponding to a set of user views V, and a sub-expression in MVG denoted *e*. The level of sharing of the sub-expression *e* in the graph MVG is a function defined as follows:

```
level_ of_sharing(e)

level = 0
for each $V_i$ in V
    for each $e_{jVi}$ in $E_{Vi}$
        if ($e \in e_{jVi}$) and ($V_i$ not marked) then
            mark $V_i$
            level = level + 1
        endif
    endfor
endfor
return (level)
end;
```

The level of sharing characterizes a sub-expression in the multi-view graph. Each expression corresponding to a user view in the multi-view graph can be characterized according to the following properties.

3.3.2. Dependency between two expressions corresponding to distinct views

Let us consider two expressions e_{iV1} and e_{jV2} representing respectively the user views V_1 and V_2 in a multi-view graph MVG. The two expressions e_{iV1} and e_{jV2} are said to be dependent if there exists a sub-expression *e* in the multi-view graph such that *e* is used in both e_{iV1} and e_{jV2}.

As a consequence, the level of sharing of the sub-expression *e* is such that:

$$\mathit{level\_of\_sharing}(e) > 0$$

In other words, two expressions representing distinct views are dependent if they have a common sub-expression *e* in the multi-view graph.

An expression e_{iVk} representing a view V_k is independent if it does not share any sub-expression with another plan e_{jVl} representing a distinct view V_l. That is, the level of sharing of each sub-expression *e* of e_{iVk} equals zero. In this case, the considered plan has no common sub-expression with any of the other plans representing a distinct view in the multi-view graph.

3.3.3. Level of dependency of an expression representing a view

The level of dependency of an expression e_{jVi} representing a user view V_i in a multi-view graph MVG is the number of distinct user views V_l such that there exists an expression e_{kVl} which share a sub-expression *e* with e_{jVi}. Let us consider the set of user views V represented by the MVG; the level of dependency of an expression e_{jVi} is a function defined as follows:

```
level_ of_dependency(eiVj)

level = 0
for each Vl not marked in V, $l \neq j$
    for each ekVl in $E_{Vk}$
        for each $e \in$ eiVj
            if ($e \in$ ekVl) then
                mark Vi
                level = level + 1
            endif
        endfor
    endfor
endfor
return (level)
end;
```

3.3.4. Redundant expressions

An expression e_{Vi} representing the user view V_i is redundant if there exists a distinct expression e'_{Vi} representing the same user view V_i and such that:

$\mathit{level\_of\_dependency}(e_{Vi}) < \mathit{level\_of\_dependency}(e'_{Vi})$

In other words, if e'_{Vi} is the expression having the highest level of dependency, then all the expressions representing the same view V_i and having a lower level of dependency are said to be redundant.

3.4. The view synthesis graph

The view synthesis graph is a canonical representation derived from the MVG. This representation is such that each user view is represented by exactly one plan. For each user view, among all the plans represented by the MVG, only one will be kept in the VSG.

One possible way to built the VSG is to use the notion of redundancy: a view synthesis graph can be defined as a set of expressions such that each user view is represented by exactly one expression, and in which each plan e_{V_i} representing the user view V_i is not redundant in the corresponding MVG. This means that the VSG is built in order to maximize the number of shared sub-expressions.

By definition and construction, the multi-view graph is unique. The view synthesis graph is not unique. For a given view V_i, there may exist more than one expression having the highest level of dependency, and consequently, the same MVG may lead to more than one VSG. The level of overlapping which characterizes a VSG is defined below.

Level of overlapping of a view synthesis graph

The level of overlapping of a view synthesis graph is the maximal level of dependency of the view expressions represented in this graph. Consider a VSG containing a number k of user views denoted V_i, each of them described by the expression denoted e_{V_i}. A level of dependency denoted n_i characterizes each expression. The level of overlapping of this VSG, denoted n(VSG) is such that:

$$n(\text{VSG}) = \text{Max}(n_i, \text{i}= 1, \text{k})$$

The VSG corresponding to a given MVG and having the highest level of overlapping must satisfy the following conditions:

- To each user view of the data warehouse is associated a single expression in the VSG.
- All the expressions in the VSG are not redundant.

The resulting VSG is a hierarchy of views. Besides the user views, this hierarchy contains intermediate views and base views defined as follows:

- Each sub-expression computed from the relations provided by a single data source is defined as a base view.
- A given sub-expression in the VSG is an intermediate view if it satisfies the following conditions:
    1. this sub-expression is shared by two or more plans in the VSG
    2. this sub-expression is neither a base view, nor a user view.

Starting from the MQG given in figure 3, the VSG which maximizes the level of overlapping is built by keeping for each view the expression having the highest level of dependency, that is by eliminating the redundant expressions. The resulting VSG is given in figure 4.

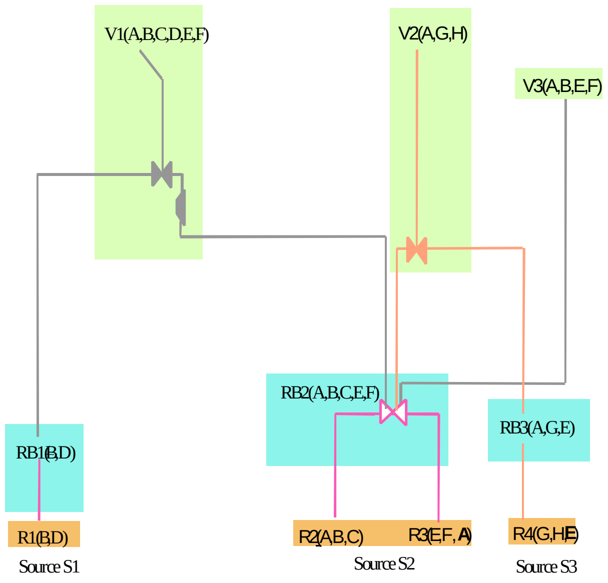

Figure 4 : Example of a logical model

Each non-redundant plan corresponding to a user view is represented by an algebraic tree. The expressions RB_1, RB_2 and RB_3 are the base views. Each of them is computed from the data provided by a single data source. These expressions can be viewed at as the extractors (or wrappers) associated with each source. The expression RB_2 is an intermediate view shared by the three user views V_1, V_2 and V_3.

4. Using the framework for data warehouse design

The key issue during the physical design of a data warehouse is to find a set of views to materialize in order to meet some user requirements. Generally, these requirements are expressed in terms of a maintenance cost, and an evaluation cost. The combination of these two costs is the operational cost of the data warehouse. The goal of the physical design is to find the set of views to materialize in order to minimize this operational cost.

Existing approaches for the selection of the views to materialize either perform an exhaustive exploration of all the possible materialization scenarios, and evaluate them to find the one that minimizes the operational cost, or use heuristics in order to reduce the number of scenarios to evaluate. In this section, we show how the proposed framework can be used for the physical design of a data warehouse. This framework allows taking into account some cost function parameters as well as some structural properties reflecting some desired quality criteria.

The considered search space is partially ordered, and may be explored using different strategies.

In the following, we first present the ordering of the search space, then the cost function parameters and the structural properties, and finally, we illustrate the use of the framework by an example of instantiation.

4.1. Ordering the search space

We consider that the data warehouse is represented by a view synthesis graph. Given the set of the user views of the data warehouse, the corresponding view synthesis graph is not unique, as we have seen in section 3. Besides this point, given a view synthesis graph, several candidate sets of materialized views may be derived, each of them representing a materialization scenario to evaluate. If we denote by VSG_i a view synthesis graph, and MS_i the sets of materialization scenarios which may be derived from the graph VSG_i, the search space for the selection of the materialized views denoted *S* is such that:
$S = MS_1 \cup MS_2 \cup .... \cup MS_n$, where n is the number of view synthesis graphs associated with the data warehouse. Therefore, exploring the search space *S* consists in considering every materialization scenario for every possible view synthesis graph representing the data warehouse.

This search space can be ordered according to two criteria:

- the level of overlapping of the view synthesis graphs representing the data warehouse,
- the number of shared expressions in the view synthesis graphs.

Given a multi-view graph, the set of view synthesis graphs having the level of overlapping p is denoted S_p, and is such that: $S_p = \{VSG^p_1, VSG^p_2, ... , VSG^p_m\}$, VSG^p_i denotes a view synthesis graph and m is the number of view synthesis graphs associated with the data warehouse and having the level of overlapping p. Each graph VSG^p_i is such that $n(VSG^p_i) = p$.

In the set S_p, the different representations can be ordered by considering subsets of representations having the same number of shared sub-expressions. Therefore, the set S_p can be partitioned in several subsets, considering the number of shared sub-expressions in the representations. A subset of S_p denoted S^n_p contains all the representations whose level of overlapping is p and whose number of shared expressions is n.

Figure 5 illustrates the partitions of the search space according to the two criteria n and p. The search space is first partitioned in several sets S_p according to the level of overlapping. Then each set S_p is partitioned in several sets S^n_p according to the number of shared expressions in the VSG.

Once the search space is ordered, it can be explored in several ways: for example, the exploration could be done starting from the representations having the highest level of overlapping, that is, representations showing the most shared expressions. These representations could then be explored according to a descending order of their number of shared sub-expressions. In our framework, the exploration strategy and the selection of the scenarios may be done according to two kinds of factors: the cost function parameters and the structural properties required for the representation of the data warehouse. Both factors are presented in sections 4.2. and 4.3.

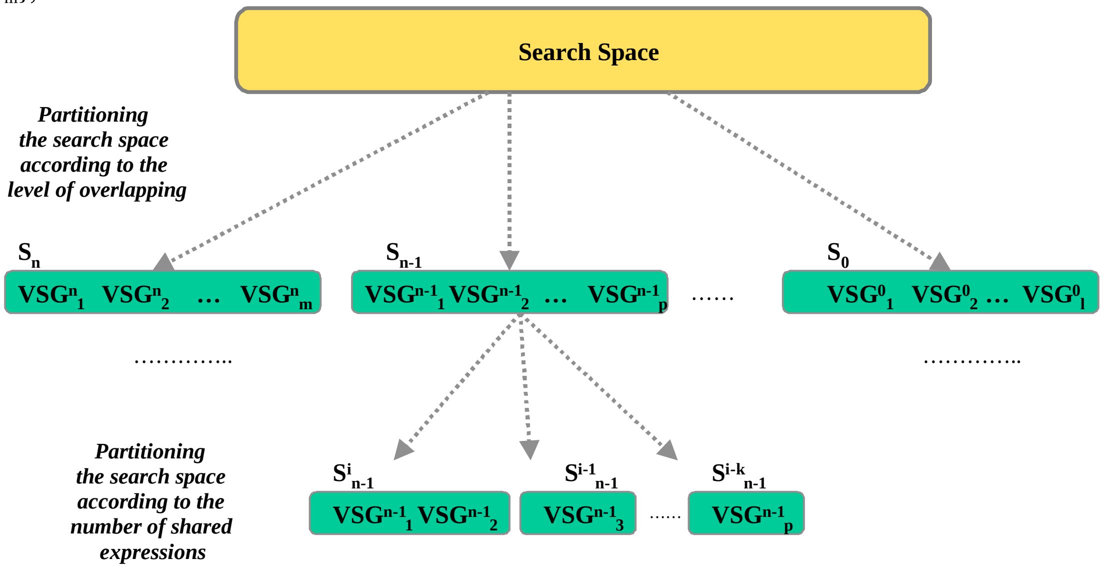
Search Space
Partitioning the search space according to the level of overlapping
S_n
VSG^n_1 VSG^n_2 … VSG^n_m
S_{n-1}
VSG^{n-1}_1 VSG^{n-1}_2 … VSG^{n-1}_p
……
S_0
VSG^0_1 VSG^0_2 … VSG^0_l
…………..
…………..
Partitioning the search space according to the number of shared expressions
S^i_{n-1}
VSG^{n-1}_1 VSG^{n-1}_2
S^{i-1}_{n-1}
VSG^{n-1}_3
……
S^{i-k}_{n-1}
VSG^{n-1}_p

Figure 5 : Partitioning the search space

4.2. The cost function parameters

The evaluation of a materialization scenario is done using a cost function which takes into account the different factors having an impact over the overall cost of the scenario. Finding such function is a domain specific problem, because the factors to consider vary from one context to another.

In the existing approaches for the selection of materialized views, the different materialization scenarios are compared by considering their respective operational cost, which is a combination of a maintenance cost and an evaluation cost. In the following, we recall the general cost function which is widely used in the literature related to the data warehouses physical design.

Each materialized view in the data warehouse has maintenance cost, which is the cost of propagating the source updates in this view. The maintenance cost of the data warehouse, denoted *M*, is the sum of the maintenance costs of all the materialized views. We denote V^m_i, i ranging from 1 to n, the materialized views of the data warehouse, f^m_i the maintenance frequency for the view V_i and C_{V_i} the maintenance cost of the view V_i; the maintenance cost of the data warehouse is such that:

$$M = \Sigma_{i=1,n} f^m_i \,.\, C_{V^m_i}$$

The user views are evaluated against the set of materialized views. They are denoted V^u_i, i ranging from 1 to p. Each of these user views has an evaluation cost denoted $E_{V^u_i}$ and a frequency denoted f^e_i. The global evaluation cost of a materialization scenario denoted *E* is such that:

$$E = \Sigma_{i=1,p} f^e_i \,.\, E_{V^u_i}$$

The operational cost of a materialization scenario denoted *OC* is a weighted sum function of the maintenance cost *M* and the evaluation cost *E*:

$$OC = E + wM$$

The best set of materialized views is the one corresponding to the minimal operational cost *OC*.

The w factor is a parameter which expresses the relative importance between the evaluation cost and the maintenance cost. Besides this factor, other parameters may be considered. For example, the users or the administrators of the data warehouse may state a value or an interval representing the acceptable operational cost of the data warehouse. These boundaries of the operational cost may be very useful to confront the cost of a scenario against the expectations of the users in terms of performance. The exploration of the search space is costly, and it is not necessary to search for an optimal solution which is far beyond the actual performance needs. The exploration may stop when a solution fitting the user requirements is found. Another possible cost function parameter could be a space limitation. All the cost function parameters are used to evaluate and select the materialization scenarios.

4.3. The structural properties

The structural properties are related to the representation of the data warehouse. They express some quality requirements specified by the user for the materialization scenario. Let us consider for example the traceability criteria. Given a set of user views, one possible requirement is to trace the intermediate results for this set in order to find the origin of an error or a misunderstanding of some aggregate value. For example, a set of aggregate values computed by different user views may be contradictory. To find the origin of the contradiction, the materialized views of the data warehouse will be checked in order to find the computation which gave raise to the error. It is obvious to see that the higher is the number of user views which use the same materialized expressions, the less is the cost of tracing a given error, because in this case, the number of materialized expressions to check is low compared to the number of expressions to consider if the users views share very few materialized expressions.

Consequently, in order to ease the tracing process, it would be more suitable to select the materialized views among the set of expressions which are used by the maximal number of user views, that is, over the view synthesis graph having the highest level of overlapping. The best materialization scenario will therefore represent a compromise between the cost parameters and the level of overlapping.

In our framework, the structural properties may be handled in the following ways:

- they may be used to constrain the selection procedure, that is to define an exploration strategy in the search space; in the example of the traceability criteria, the search space may be explored exhaustively starting from the representations having the highest level of overlapping,
- they may be used to reduce the search space, by selecting a sub-set of representations satisfying them,
- and finally, they may be used after the determination of the scenario which best fits the cost function parameters, in order to check whether this scenario conforms to the structural properties or not.

4.4. An instantiation of the general framework

We will now present an example of selection of a materialization scenario considering, additionally to the ordering criteria of the search space, the following factors: first, the users or the administrators of the data warehouse have stated an interval representing the acceptable operating cost of the data warehouse; second, some traceability requirements have been expressed by the users. These requirements will be used to constrain the selection procedure.

If we consider that the user requirements in terms of performance are expressed by the interval $[l_1, l_2]$, where l_1 and l_2 represents the lowest and the highest acceptable operational cost respectively, the exploration of the ordered search space will stop if the operational cost *OC* corresponding to a materialization scenario ms_i is such that:

$$l_1 \leq OC(ms_i) \leq l_2.$$

Considering the traceability criteria, the approach for the selection of the views to materialize will focus on the shared sub-expressions, and choose the materialized views in the set of sub-expressions shared by the maximal number of views. The exploration strategy is then to consider first representations having the highest level of overlapping. The instantiation of our general framework considering these requirements is given in figure 6.

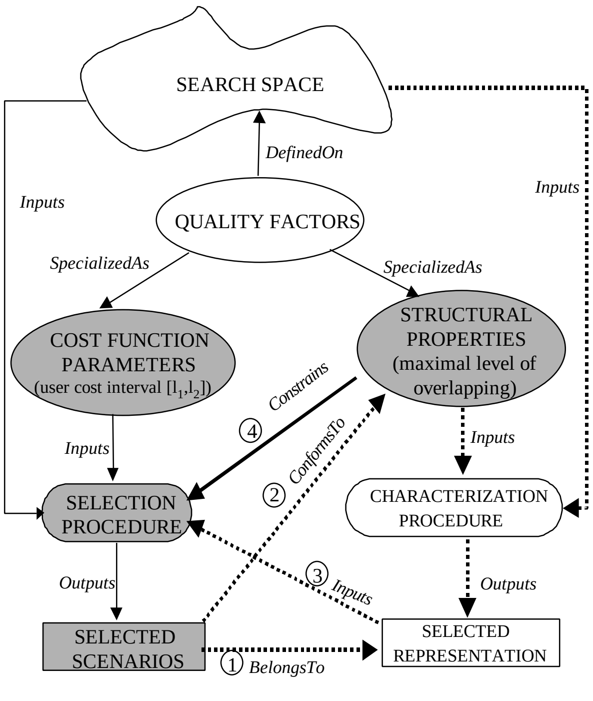

Figure 6 : Instantiation of the general framework

The characterization procedure is not performed, and the paths used in the instantiation are represented by the continuous lines. If p is the maximal level of overlapping which can be derived from the multi-view graph, then the corresponding set S_p contains the sub-expressions which are shared by the maximal number of user views. These sub-expressions will be first considered when determining the views to materialize. Multiple materialization scenarios can be generated from a single graph VSG^p_i. Each of them should be evaluated. After considering the graphs of a set S_p, the graphs of the set S_{p-1} are considered, and the cost of the materialization scenarios is evaluated for each graph VSG^{p-1}_i. This process iterates until a solution is found within the boundaries specified by the users.

This exploration of the search space starting from the representations having the maximal level of overlapping is described by the following general algorithm which uses the following notations:

- MVG is the multi-view graph describing the data warehouse,
- S_p the set of view synthesis graph derived from MVG and having the level of overlapping p,
- p_{max} the maximal level of overlapping of the expressions in MVG,
- VSG^p_i a view synthesis graph whose level of overlapping is p,
- $[l_1, l_2]$ the interval representing the user requirements, that is the interval in which the operational cost varies.
- MS_p the set of materialization scenarios derived from the view synthesis graphs whose level of overlapping is p,
- ms_i a materialization scenario,
- $OC(ms_i)$ the operational cost of the materialization scenario ms_i.

```
Exploration procedure
p = $p_{max}$ ;
Generate $S_p$ from MVG;
While (solution not found)
If  $S_p \neq \emptyset$ Then
    For each $VSG^p_i$ in $S_p$
        Generate $MS_p$
        For each $ms_i$ in $MS_p$
            If $l_1 \leq OC(ms_i) \leq l_2$
            Then  ($ms_i$) is an acceptable solution;
            Endif
        Endfor
    Endfor
Endif
p = p-1;
Generate $S_p$ from MVG;
Endwhile
End
```

We did not present in this paper the details of the selection of the candidate sets of views to materialize; each set of candidate materialized views should allow to compute all the user views, that is, each user view can be rewritten using the set of materialized views.

5. Related works

Several approaches have been proposed for the physical design of a data warehouse. The problem addressed by these approaches is to find a set of views to materialize in order to meet the performance requirements. These performance requirements are expressed as an operational cost, which is a combination of an evaluation cost and a maintenance cost. Some of the approaches dealing with this design problem are described in [Baralis et al 97], [Theodoratos & Sellis 97], [Theodoratos & Sellis 98], [Theodoratos et al 99], [Ligoudistianos et al 99] and [Yang et al 97].

Two kinds of solutions are generally proposed for the selection of the views to materialize in a data warehouse, either solutions consisting in an exhaustive exploration of the possible materialization solutions to find to optimal one, or solutions based on some heuristics which reduce the exploration. In [Yang et al 97], two algorithms for selecting the views to materialize are proposed; the first one is an exhaustive exploration of all the possible plans of the users queries to find an optimal multiple view processing plan, the second one is an heuristic algorithm which considers only the optimal plan for each user query.

In [Baralis et al 97], the proposed algorithm addresses the materialized view selection problem in the context of multidimensional databases. [Theodoratos & Sellis 97] and [Theodoratos & Sellis 98] present the view selection problem as a state space problem. In [Theodoratos et al 99], a pruning algorithm and a greedy algorithm are presented to select the set of views to materialize. But the heuristics used do not guarantee to find the optimal solution. [Ligoudistianos et al 99] proposes an A* algorithm which delivers the optimal solution, and presents some heuristics to prune the considered state space.

In this paper, rather than a new approach for the selection of the best set of materialized views, we tried to define the main features of a framework for data warehouse physical design. This physical design is done on a search space consisting of a set of representations of the data warehouse, considering the level of overlapping and the number of shared expressions which characterize each representation. This level allows to order the search space, and the exploration is done according to a strategy which may be driven by the user requirements. These requirements may be either some cost function parameters or some structural properties. Existing approaches introduce cost function parameters such as space limitation, but they do not integrate the structural properties which express some quality factors such as evolutivity, security or traceability.

6. Conclusion

In this paper we have presented an extensible framework for data warehouse physical design. This framework is based on a logical schema, which is a canonical representation of the data warehouse views. This logical model is built from subsets of the possible plans for each user view, which are integrated in an intermediate representation called the multi-view graph. The logical model, called the view synthesis graph (VSG), is derived from the multi-view graph. A level of overlapping characterizes this model. It represents the maximal number of views sharing the same expression in the graph.

The framework for physical design uses the VSG as a representation of the views of a data warehouse. Since this VSG is not unique, the search space for the selection of the views to materialize is composed of all the scenarios derived from all the possible VSG. This search space can be ordered considering the level of overlapping and the number of shared expressions of the representations from which the scenarios are derived.

This ordered search space can be explored in different ways; one possible exploration strategy consists in giving priority to the most shared sub-expressions during the selection of the views to materialize, and therefore exploring first the representations having the highest level of overlapping. The exploration is guided by some users requirements, which can either be related to the cost function, such as the upper an lower bounds representing acceptable operational cost, or related to some structural properties of the logical model, such as evolutivity.

Further works will consist in identifying sets of quality factors related to the physical design of the data warehouses, and study the impact of these factors on both the selection and the characterization procedures of the proposed framework.

References

[Baralis et al 97] E. Baralis, S. Paraboschi, E. Teniente. Materialized View Selection in a Multidimensional Environment. In *Proc. of the 23rd International Conference on Very Large Data Bases (VLDB'97),* Athens, Greece, August 1997.

[Bouzeghoub et al 99] Bouzeghoub M., Fabret F., Matulovic M., Modeling the Data Warehouse Refreshment Process as a Workflow Application. *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, Heidelberg, Germany, June 1999.

[Bouzeghoub & Theodoratos 99] Bouzeghoub M., Theodoratos D. Data Currency Quality Factors in Data Warehouse Design. Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg, Germany, June 1999.

[Calvanese et al 98] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati. Source Integration integration : conceptual modeling and reasoning support. In *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98)*, New York City, USA, August 1998.

[Chawathe et al 94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom. The TSIMMIS project : Integration of Hererogeneous Information Sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94)*, Tokyo, Japan, October 1994.

[Chaudhuri & Dayal 96] S. Chaudhuri, A. Dayal. An Overview of Data Warehousing and OLAP Technology. Tutorial, In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB'96)*, Mumbai, India, September 1996.

[Garcia-Molina et al 95] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajamaran, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom. The TSIMMIS approach to mediation: Data model and Languages. In *2nd International Workshop on Next Generation Information Technologies and Systems (NGITS '95)*, Naharia, Israel, June 1995.

[Gupta & Mumick 95] A. Gupta, I.S. Mumick, Maintenance of materialized views: Problems, Techniques, and applications. In *IEEE Data Engineering Bulletin, Special Issue on Materialised Views and Data Warehousing*, 18(2), june 1995

[Hammer et al 95] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, Y. Zhuge. The Stanford Data Warehousing Project. *IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing*, 18(2), June 1995

[Hull & Zhou 96] R. Hull, G. Zhou. A Framework for Supporting Data Integration Using the Materialized and Virtual Approches. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, June 1996.

[Jarke et al 97] M. Jarke, Y. Vassiliou. Foundation of Data Warehouse Quality: an Overview of the DWQ project . In *Proceedings of the 2nd Internat. Conf. on Information Quality*, Cambridge, Mass, 1997.

[Jarke et al 99] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis. (edits). Fundamentals of Data Warehouses. Springer, 1999.

[Jarke et al 99a] Jarke M, Jeusfeld M.A., Quix C., Vassiliadis P., "Architecture and Quality in Data Warehouses", Proceeding of the 10th International Conference on Advanced Information Systems Engineering (CAiSE'98), Pisa, Italy, June 1998.

[Jeusfeld et al 98] Jeusfeld, M. A., Quix C., Jarke M., "Design and Analysis of Quality Information for Data Warehouses", Proceedings of the 17th Internat. Conf. on Conceptual Modeling (ER'98), Singapore, november 1998.

[Kedad 99] Z. Kedad. Techniques d'intégration dans les systèmes d'information multi-source. *PhD Thesis*, Laboratoire PriSM, University of Versailles, France, January 1999.

[Kedad & Bouzeghoub 99] Z. Kedad, M. Bouzeghoub. Conception de systèmes d'information multi-source. In *Proceedings of INFORSID'99*, Toulon, France, June 1999.

[Levy et al 96] A.Y. Levy, A. Rajaraman, J.J. Ordille. Query-Answering Algorithms for Information Agents. In Proceedings of the 13th National Conference on Artificial Intelligence and 8th Innovative Applications of Artificial Intelligence Conference (AAAI 96, IAAI 96), Portland, Oregon, August 1996.

[Levy et al 96a] A.Y. Levy, A. Rajaraman, J.J. Ordille. Querying Heterogeneous Information Sources Using Source Description. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB'96)*, Mumbai, India, September 1996.

[Ligoudistianos et al 99] S. Ligoudistianos, T. Sellis, D. Theodoratos, Y. Vassiliou. Heuristic Algorithms For Designing The Data Warehouse with SPJ Views. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'99)*, 1999.

[Theodoratos & Sellis 97] D. Theodoratos, T. Sellis. The Data Warehouse Configuration. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97),* Athens, Greece, August 1997.

[Theodoratos & Sellis 98] D. Theodoratos, T. Sellis. Data Warehouse Schema and Instance Design. In *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98),* Singapore, November 1998.

[Theodoratos et al 99] D. Theodoratos, S. Ligoudistianos, T. Sellis. Designing the Global Data Warehouse with SPJ Views. In *Proceedings of the 11th Conference on Advanced Information Systems Engineering (CAISE'99),* Heidelberg, Germany, June 1999.

[Vassiliadis et al 99] P. Vassiliadis, M. Bouzeghoub, C. Quix. Towards quality-oriented data warehouse usage and evolution. *Proceedings of the 11th Conference on Advanced Information Systems Engineering (CAiSE'99),* Heidelberg, Germany, June 1999.

[Widom 95] J. Widom. Research Problems in Data Warehousing. In *Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95),* Baltimore, Maryland, November 1995.

[Wierner et al 96] J.L. Wiener, H. Gupta, W.J. Labio, Y. Zhuge, H. Garcia-Molina, J. Widom. A System Prototype for Warehouse View Maintenance. In VIEWS 96

[Yang et al 97] J. Yang, K. Karlapalem, S. Li. Algorithms for materialized view design in data warehousing Environment. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97),* Athens, Greece, August 1997.

[Zhou et al 95] G. Zhou, R. Hull, R. King, J.C. Franchitti. Data Integration and Warehousing using H2O. *IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing*, 18(2), June 1995