Comparing Nested GPSJ Queries
in Multidimensional Databases

Matteo Golfarelli
DEIS – University of Bologna
Viale Risorgimento, 2
40136 Bologna - Italy
+39-0547-642862

golfare@csr.unibo.it

Stefano Rizzi
DEIS – University of Bologna
Viale Risorgimento, 2
40136 Bologna - Italy
+39-051-2093542

srizzi@deis.unibo.it

ABSTRACT

A multidimensional database can be seen as a collection of multidimensional cubes, from which information is usually extracted by aggregation; aggregated data can be calculated either from cubes containing elemental data or from views in which partially aggregated data are stored. Thus, view materialization and run-time optimization through query rewriting become crucial issues in determining the overall performance. The capability of matching two queries is necessary to address both issues; unfortunately, most works in this field consider only simple categories of queries. In this paper we focus on a relevant class of queries, those modeled by Nested Generalized Projection / Selection / Join (NGPSJ) expressions, in which different aggregation functions may be applied in sequence to the same measure and selections may be formulated, at different granularities, on both dimensions and measures of the cube. Given two NGPSJ expressions, we show how to recursively compute their ancestor, i.e., the coarsest expression on which both can be rewritten. The ancestor has a key role in view materialization, since it may be used to determine a restricted set of candidate views; given the ancestor, determining if one expression can be rewritten on the other is straightforward.

1. INTRODUCTION

Multidimensional databases have gathered wide research and market interest as the core of decision support applications such as data warehouses. A *multidimensional database* (MD) can be seen as a collection of multidimensional cubes centered on facts of interest (for instance, the sales in a chain store); within a cube, each cell contains information useful for the decision process, i.e., a set of numerical *measures*, while each axis represents a possible *dimension* for analysis. Typically, for each dimension, a hierarchy of aggregation levels is defined.

The basic mechanism to extract useful information from elemental data in MDs is aggregation. Within typical multidimensional queries, the values of measures are summarized according to some combination of aggregation levels from the different dimensions, which defines the coarseness of aggregation. The aggregated data which solve the query can be calculated either from cubes containing elemental data or from cubes where partially aggregated data are pre-calculated and stored (*views*). Thus, both the problem of deciding which views should be materialized and that of finding if and how a given query can be rewritten on a view (i.e., it can be answered using that view) assume a crucial role in determining the system overall performance.

The capability of matching two queries is necessary in the context of both view materialization [1][6][7] and query rewriting aimed at run-time optimization [2][8]. Unfortunately, most works consider only simple categories of queries; to the best of our knowledge, only one approach [9] considers both query nesting and multidimensional aggregation, and it pragmatically proposes a list of patterns for rewriting multi-block SQL queries on views.

In the MD context, a particularly interesting and general class of queries are those represented by Nested Generalized Projection / Selection / Join (NGPSJ) expressions, in which different aggregation functions may be applied in sequence to the same measure and selections may be formulated, at different granularities, on both dimensions and measures to properly restrict the data to be aggregated. In [3] we introduced this class of queries and proposed an original approach to materialization in which views themselves are defined by NGPSJ expressions.

Typically, the first step in view materialization consists in determining a restricted set of *candidate views*, i.e., views which are potentially useful for query answering. In [1], the authors found their definition of candidate views on an *ancestor* operator which, given two queries, returns the smallest view on which both can be answered. Computing the ancestor in their approach is straightforward, since queries are characterized only by the coarseness of aggregation.

In this paper we show how to compute the ancestor between two NGPSJ expressions. The nested structure of NGPSJ expressions suggests to execute matching in a piece-by-piece fashion; thus, we propose a set of incremental rules and show how their recursive application leads to computing the ancestor.

Given two NGPSJ expressions and their ancestor, determining if one expression can be rewritten on the other is straightforward. In this paper we do not specifically address the problem of actually determining the rewrite of an

expression on another due to the lack of space; however, the rules presented here can be easily extended to recursively determine also the rewriting of the two source expressions on their ancestor.

The paper is organized as follows. In Section 2 we present our working example and demonstrate the utility of NGPSJ modeling with an example. After introducing the concept of pattern in Section 3 and defining NGPSJ expressions in Section 4, in Section 5 we define the relationship of rewritability between expressions, introduce the rules for determining the ancestor and show how they can be applied.

2. WORKING EXAMPLE

An MD implemented on a relational DBMS is usually organized according to the so-called *star scheme*, in which each cube is represented by one *fact table* storing the measures and one denormalized *dimension table* for each dimension. The primary key of each dimension table (usually a *surrogate key*, i.e., internally generated) is imported into the fact table; the primary key of the fact table is defined by the set of these foreign keys. Each dimension table contains a set of *attributes* defining a hierarchy of aggregation levels.

The simple *Sales* star scheme used as a working example is defined below:

STORE (StoreId, SName, SCity)
TIME (TimeId, TDay, TMonth, TYear)
PRODUCT (ProductId, PName, PType, PCategory)
SALE (TimeId, StoreId, ProductId, Qty, Prc)

The denormalized dimension tables leave out the following hierarchies of functional dependencies:

$$SName \rightarrow SCity, TDay \rightarrow TMonth \rightarrow TYear,$$
$$PName \rightarrow PType \rightarrow PCategory$$

In the rest of the paper, we denote with T the join between the fact and the dimension tables: T = SALE ⋈ PRODUCT ⋈ STORE ⋈ TIME.

Given a star scheme S, we denote with $Patt(S)$ the set of key attributes of the fact table; we denote with $Attr(S)$ and $Meas(S)$ the sets of non-key attributes of, respectively, the dimension tables and the fact table:

$Patt(Sales) = \{SName, TDay, PName\}$, $Meas(Sales) = \{Qty, Prc\}$,
$Attr(Sales) = \{ SName, SCity, TDay, TMonth, TYear, PName, PType, Pcategory\}$

In order to demonstrate the utility of using NGPSJ expressions to model queries and views, consider the following queries, belonging to the workload on the *Sales* star scheme:

q': "*For each product, find the average selling price and the maximum total quantity sold for the stores which sold more then 1000 units of that product*"

q'': "*For each product of type beverage and for each year, find the total quantity sold and the average of the total monthly revenues that are higher than 1000*"

In Section 4 we will show how these queries can be modeled by NGPSJ expressions. Among the possible views defined according to the approach proposed in [1], none could be used to answer either q' or q'': in fact, within such views, Qty and Prc would be aggregated only by means of the sum and the average

operators, respectively. Thus, both queries would be necessarily answered on the base fact table, with a high execution cost. On the other hand, if NGPSJ views were materialized, new possibilities for on-line query optimization would arise. For instance, let a view v be defined as follows:

v: "*For each product, store and month, store the total quantity sold, the average price, the total revenue and the number of sales*"

Both q' and q'' can be answered on it, significantly decreasing the execution costs.

3. MODELING PATTERNS

The concept of aggregation pattern plays a basic role in multidimensional databases since it allows different levels of aggregation to be modeled:

Definition 1. Given a star scheme S, an *aggregation pattern* (or simply *pattern*) on S is a set $P \subset Attr(S)$ such that no functional dependency exists between each pair of attributes in P: $\forall a_i \in P (\nexists a_j \in P \mid a_i \rightarrow a_j)$.

Definition 2. Given two patterns P' and P'' on scheme S, we say that P' is *coarser* than P'' ($P' \leq P''$) iff $\forall a_i \in P' (a_i \in P'') \vee (\exists a_j \in P'' \mid a_j \rightarrow a_i)$.

Manipulating patterns also requires, given two patterns, to determine their *ancestor*:

Definition 3. Given two patterns P' and P'' on scheme S, the *ancestor* of P' and P'' is the pattern $P' \overset{P}{\oplus} P''$ such that (1) ($P' \leq P' \overset{P}{\oplus} P''$) ∧ ($P'' \leq P' \overset{P}{\oplus} P''$) and (2) for each other pattern P which satisfies (1), $P' \overset{P}{\oplus} P'' \leq P$ holds.

Theorem 1. Given two patterns P' and P'' on scheme S, $P' \overset{P}{\oplus} P'' = \left\{ a_i \in P' \cup P'' \mid \nexists a_j \in P' \cup P'', a_j \neq a_i \mid a_j \rightarrow a_i \right\}$.

Proof. Let $P^* = \left\{ a_i \in P' \cup P'' \mid \nexists a_j \in P' \cup P'', a_j \neq a_i \mid a_j \rightarrow a_i \right\}$. Proving that P^* is the ancestor requires proving that (a) $P' \leq P^*$ ∧ $P'' \leq P^*$; and that (b) $\forall P \mid P' \leq P \wedge P'' \leq P (P^* \leq P)$.

(a) Applying Definition 2 we find

$$P' \leq P^* \Leftrightarrow \forall a_i \in P' (\exists a_j \in P^* \mid a_j \rightarrow a_i) \vee (a_i \in P^*) \quad (1)$$

Let $a_i \in P'$; either $a_i \in P^*$ or $a_i \notin P^*$ hold. In the first case, (1) is proved. In the second, necessarily $\exists a_j \in P' \cup P'' \mid a_j \rightarrow a_i$. For Definition 1, $a_j \notin P'$ hence $a_j \in P''$. Thus, $\nexists a_k \in P' \cup P'' \mid a_k \rightarrow a_j$, which means that $a_j \in P^*$ so (1) is proved. Similarly for P''.

(b) Suppose ab absurdo that $\exists \overline{P} \mid P' \leq \overline{P} \wedge P'' \leq \overline{P} \wedge P^* \npreceq \overline{P}$. $P^* \npreceq \overline{P}$ holds iff

$$\exists \overline{a}_i \in P^* \mid \left(\forall a_j \in \overline{P} (a_j \nrightarrow \overline{a}_i) \right) \wedge \left(\overline{a}_i \notin \overline{P} \right) \quad (2)$$

If $\overline{a}_i \in P^*$ then either $\overline{a}_i \in P'$ or $\overline{a}_i \in P''$. In the first case, since $P' \leq \overline{P}$ then $\forall a_i \in P' (\exists a_j \in \overline{P} \mid a_j \rightarrow a_i) \vee (a_i \in \overline{P})$; thus, either

$\overline{a}_i \in \overline{P}$ or $\exists \overline{a}_j \in \overline{P} \mid \overline{a}_j \rightarrow \overline{a}_i$, which contradicts (2). Similarly if $\overline{a}_i \in P''$. ❑

It is straightforward to verify that the ancestor operator is reflexive, commutative and associative.

Example 1. On the *Sales* scheme, example of patterns are $P' = \{SCity,TMonth,PType\}$, $P'' = \{SCity,TYear\}$, $P''' = \{SCity,TDay\}$. In Figure 1 the relationships between these patterns are shown; an arrow from one pattern to another denotes that the second is coarser than the first: $P'' < P'$, $P'' < P'''$, $P' \overset{P}{\oplus} P''' = \{SCity,TDay,PType\}$. ❑
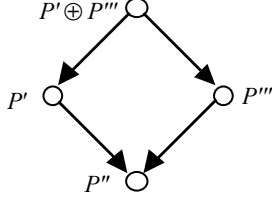

Figure 1. Coarseness of patterns.

4. MODELING NGPSJ EXPRESSIONS

A GPSJ (Generalized Projection / Selection / Join) expression [5] is a selection σ_1 over a generalized projection π over a selection σ_2 over a set of joins χ: $\sigma_1\pi\sigma_2\chi$. The *generalized projection* operator, $\pi_{P,M}(R)$, is an extension of duplicate eliminating projection, where P denotes a pattern and M denotes a set of *aggregate measures*, each defined by applying an aggregate function to an algebraic expression involving attributes in R.

Nesting GPSJ expressions means using the result from an expression as the input for another; it adds expressive power to GPSJ expressions since:

- Sequences of aggregate functions may be applied to a measure (e.g., the maximum of the sums);

- Multiple selections can be applied to measures at different granularities.

On the other hand, nesting requires issues related to partially aggregated data to be addressed: in particular, some aggregations require support measures in order to be correctly calculated from partial aggregates (e.g., the average function requires the cardinality of each partial aggregate).

In this paper we consider NGPSJ expressions applied to star schemes. We define them by induction, by supplying four construction rules; each expression e is described by its outer pattern, $ExtP(e)$, the aggregate measures returned, $ExtM(e)$, and the outer selection predicate on measures, $ExtS(e)$.

Construction Rule A. Given a star scheme S with base fact table FT and dimension tables $DT_1,...DT_n$, $e = FT \bowtie DT_1...\bowtie DT_n$ is an NGPSJ expression, described by $ExtP(e) = Patt(S)$, $ExtM(e) = Meas(S)$ and $ExtS(e) = TRUE$.

Construction Rule B. If e is an NGPSJ expression, and given a predicate $PredA$ expressed as a conjunction of Boolean predicates each involving one attribute in $Attr(S)$, $e' = \sigma_{PredA}(e)$

is an NGPSJ expression, described by $ExtP(e') = ExtP(e)$, $ExtM(e') = ExtM(e)$ and $ExtS(e') = ExtS(e)$.

Construction Rule C. If e is an NGPSJ expression, and given a pattern $P < ExtP(e)$ and a set M of aggregate measures, each defined by applying an aggregate function to algebraic expressions involving measures in $ExtM(e)$, $e' = \pi_{P,M}(e)$ is an NGPSJ expression, described by $ExtP(e') = P$, $ExtM(e') = M$ and $ExtS(e') = TRUE$.

Construction Rule D. If e is an NGPSJ expression, and given a predicate $PredM$ expressed as a disjunction of conjunctions of simple Boolean predicates on measures in $ExtM(e)$, $e' = \sigma_{PredM}(e)$ is an NGPSJ expression, described by $ExtP(e') = ExtP(e)$, $ExtM(e') = ExtM(e)$ and $ExtS(e') = PredM$.

We say an NGPSJ expression is in *normal form* when (1) selections on attributes are pushed below all projections and selections on measures[1] and (2) projection and selections are coalesced as much as possible. Within the normal form, no two consecutive selections on measures can be present; on the other hand, two consecutive projections may be necessary to allow two or more different aggregate functions to be applied in sequence on a measure.

We wish to emphasize that the order in which operators appear within the expressions *does not* reflect the execution plans that will be used to calculate them. Besides, we assume for simplicity that the fact table is always joined to *all* the dimension tables (though in some expressions, depending on the pattern required, it may be possible to omit one or more of them).

Example 2. Consider the following queries on the *Sales* star scheme:

q': "*For each product, find the average selling price and the maximum total quantity sold for the stores which sold more then 1000 units of that product*"

q'': "*For each product of type beverage and for each year, find the total quantity sold and the average of the total monthly revenues that are higher than 1000*"

These queries are expressed, respectively, by the following NGPSJ expressions in normal form:

$$e' = \pi_{PName,WAVG(P,C),MAX(Q)}\sigma_{Q>1000}$$
$$\pi_{PName,SName,Q=SUM(Qty),P=AVG(Prc),C=COUNT(*)}(T)$$

$$e'' = \sigma_{A>1000}\pi_{TYear,PName,SUM(Q),A=AVG(R)}\pi_{TMonth,PName,Q=SUM(Qty),R=SUM(Qty.Prc)}$$
$$\sigma_{PType='Bev'}(T)$$

where $WAVG(m,w)$ computes the weighted average of measure m based on the weights w. These expressions are characterized by the following properties and SQL formulations:

$$ExtP(e') = \{PName\},\ ExtP(e'') = \{TYear,PName\},$$
$$ExtM(e') = \{WAVG(P,C),MAX(Q)\},\ ExtM(e'') = \{AVG(R),SUM(Q)\},$$
$$ExtS(e') = TRUE,\ ExtS(e'') = (A>1000).$$

---

[1] While in general moving a selection on a measure in different positions of an expression affects the expression semantics, selections on attributes can be placed indifferently in any position outside the joins.

```
e': SELECT PN,SUM(P*C)/SUM(C),MAX(Q)
    FROM (   SELECT  PRODUCT.PName AS PN,
                     STORE.SName AS SN,
                     SUM(SALE.Qty) AS Q,
                     AVG(SALE.Prc) AS P,
                     COUNT(*) AS C
             FROM SALE,PRODUCT,STORE
             WHERE <...join conditions...>
             GROUP BY PN,SN)
    WHERE Q>1000
    GROUP BY PN

e": SELECT PN,TY,SUM(Q),AVG(R) AS A
    FROM (   SELECT  PRODUCT.PName AS PN,
                     TIME.TMonth AS TM,
                     TIME.TYear AS TY,
                     SUM(SALE.Qty) AS Q,
                     SUM(SALE.Prc*SALE.Qty) AS R
             FROM SALE,PRODUCT,STORE
             WHERE <...join conditions...>
             AND PRODUCT.PType='Bev'
             GROUP BY PN,TM)
    GROUP BY PN,TY
    HAVING A>1000
❏
```

4.1 Aggregate functions

An aggregate function maps a multiset of values into a single value. A classification of aggregate functions which is relevant to our approach is presented in [4]:

- *Distributive*, that allows aggregates to be computed directly from partial aggregates.

- *Algebraic*, that require additional information (*support measures*) to calculate aggregates from partial aggregates.

- *Holistic*, that do not allow aggregates to be computed from partial aggregates through a finite number of support measures.

Consider aggregating a two-dimensional set of values $\{x_{ij} \mid i=1,\dots I, \ j=1,\dots J\}$. As stated in [4], for a distributive or algebraic aggregate function f, it is always possible to find an aggregate function h and a finite set of aggregate functions $\{g_k \mid k=1,\dots K\}$ such that $f(\{x_{ij}\}) = h(\{g_1(\{x_{ij} \mid i=1,\dots I\}) \mid j=1,\dots J\},\dots\{g_K(\{x_{ij} \mid i=1,\dots I\}) \mid j=1,\dots J\})$. If f is distributive, $K=1$ and $g \equiv f$. Table I reports the g and h functions associated to some common aggregation functions.

Table I. The g and h functions associated to some common aggregation functions f.

f	g	h
SUM(m)	$m' = $ SUM(m)	SUM(m')
MAX(m)	$m' = $ MAX(m)	MAX(m')
COUNT(*)	$c = $ COUNT(*)	SUM(c)
AVG(m)	$m' = $ AVG(m), $c = $ COUNT(*)	WAVG(m',c)
WAVG(m,c)	$m' = $ WAVG(m,c), $c' = $ COUNT(*)	WAVG(m',c')

Given a set M of aggregate measures, we denote with $G(M)$ and $H(M)$ the sets of aggregate measures obtained by substituting each aggregate function f in M with its corresponding g function(s) and h function, respectively.

5. COMPARING NGPSJ EXPRESSIONS

Two NGPSJ expressions e' and e'' on star scheme S are equivalent ($e' = e''$) when they return the same result for every legal instance of S, i.e., each column returned by an expression matches with an equivalent column in the other. Among the columns returned, testing the equivalence of those included in the outer pattern is straightforward, since they correspond to attributes in $Attr(S)$. On the other hand, the aggregate measures returned are not base measures belonging to $Meas(S)$; they are computed from other measures which, in turn, may be defined within a nested sub-expression. For this reason, the process of comparing two NGPSJ expressions is inherently recursive.

Definition 4. Given two NGPSJ expressions e' and e'' on scheme S, we say that e' is *rewritable* on e'' ($e' \leq e''$) if, by repeatedly applying Construction Rules B, C and D to e'', it is possible to obtain a NGPSJ expression which is equivalent to e'.

Intuitively, rewritability holds when every column returned by e' can be computed from those returned by e''.

Example 3. Expression $\pi_{PType,TYear,SUM(Q)}\sigma_{Q>2000}\pi_{PName,TMonth,Q=SUM(Qty)}$ $\sigma_{PCategory='Foodstuffs' \wedge TYear>'1997'}(T)$ is rewritable on

$$e = \sigma_{Q>1000}\pi_{PName,TMonth,Q=SUM(Qty)}(T)$$

by computing $\pi_{PType,TYear,SUM(Q)}\sigma_{Q>2000}\sigma_{PCategory='Foodstuffs' \wedge TYear>'1997'}(e)$. On the other hand, expression $\sigma_{Q>1000}\pi_{PName,TMonth,Q=SUM(Qty)}\sigma_{Prc>10}$ $\sigma_{TDay>'Feb15,97'}(T)$ is not rewritable on e since it specifies additional selections on an attribute which did not appear in e due to aggregation (TDay) and on a measure which in e is not available at the granularity required (Prc). ❏

Definition 5. Given two NGPSJ expressions e' and e'' defined on scheme S, the *ancestor* of e' and e'' is the NGPSJ expression $e' \oplus e''$ such that (1) $(e' \leq e' \oplus e'') \wedge (e'' \leq e' \oplus e'')$ and (2) for each other NGPSJ expression e which satisfies (1), $e' \oplus e'' \leq e$ holds.

Intuitively, the ancestor between two expressions is the coarsest expression on which both can be rewritten. In some view materialization techniques [1][3], this operator is necessary since every expression which is ancestor of at least two queries in the workload defines a candidate view.

Given two NGPSJ expressions, one and only one ancestor in normal form always exists.

Theorem 2. The ancestor operator is reflexive, commutative and associative.

Proof. Proving reflexivity and commutativity is straightforward. Associativity holds iff both the following relationships hold:

$$(e' \oplus e'') \oplus e''' \leq e' \oplus (e'' \oplus e'''), \quad e' \oplus (e'' \oplus e''') \leq (e' \oplus e'') \oplus e'''$$

Let us consider the first relationship. Applying part (1) of Definition 5 to the rightmost expression, we find $e' \leq e' \oplus (e'' \oplus e''')$, $e'' \leq e'' \oplus e''' \leq e' \oplus (e'' \oplus e''')$ and $e''' \leq e'' \oplus e''' \leq e' \oplus (e'' \oplus e''')$. Thus, due to part (2) of Definition 5, it must be $e' \oplus e'' \leq e' \oplus (e'' \oplus e''')$. Again, due to part (2) of Definition 5, we can

write $(e' \oplus e'') \oplus e''' \leq e' \oplus (e'' \oplus e''')$. The second relationship can be symmetrically proved. ❏

The possible relationships between two NGPSJ expressions e' and e'' are sketched in Figure 2:

- Case A: $e' = e'' \Leftrightarrow e' \oplus e'' = e' = e''$

- Case B: $e' \leq e'' \Leftrightarrow e' \oplus e'' = e''$

- Case C: $e'' \leq e' \Leftrightarrow e' \oplus e'' = e'$

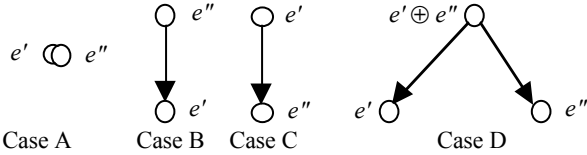- Case D: $e' \nleq e'' \wedge e'' \nleq e' \Leftrightarrow (e' < e' \oplus e'') \wedge (e'' < e' \oplus e'')$

Figure 2. Different relationships between two NGPSJ expressions; an arrow from one expression to another denotes that the second is rewritable on the first.

In the following subsections we introduce a set of rules which can be used to recursively determine the ancestor of two given NGPSJ expressions. It is straightforward to prove that these rules satisfy part (1) of Definition 5. We are currently working to formally prove that they also satisfy part (2); however, the large amount of sample expressions examined so far revealed no counter-examples.

5.1 Predicates on attributes
Rule 1. Given two NGPSJ expressions

$$e' = \sigma_{PredA'}(FT \bowtie DT_1 \ldots \bowtie DT_n), \ e'' = \sigma_{PredA''}(FT \bowtie DT_1 \ldots \bowtie DT_n)$$

(where possibly $PredA' = TRUE$ and/or $PredA'' = TRUE$), it is

$$e' \oplus e'' = \sigma_{PredA}(FT \bowtie DT_1 \ldots \bowtie DT_n)$$

where $PredA$ is a conjunction of Boolean predicates, each involving one attribute $a_i \in Attr(S)$ and expressed as the disjunction of the two predicates involving a_i in $PredA'$ and $PredA''$, respectively.

Example 4. Given $e' = \sigma_{PCategory='x'}(T)$ and $e'' = \sigma_{PCategory \ IN \ ('x','y','z') \ \wedge \ TYear<1999}(T)$, it is $e' \oplus e'' = \sigma_{((PCategory='x') \ \vee \ (PCategory \ IN \ ('x','y','z')) \ \wedge \ (TRUE \ \vee \ (TYear<1999))}(T) = \sigma_{PCategory \ IN \ ('x','y','z')}(T)$. ❏

5.2 Generalized projections
Rule 1 solves predicates on attributes by relaxing the most restrictive condition for each attribute in the two source expressions. When considering generalized projections, it is necessary to consider the attributes involved in predicates that have been relaxed: in fact, the external pattern of the ancestor must be fine enough to allow the relaxed predicates to be applied when rewriting the source expressions. Thus, given two predicates on attributes, $PredA'$ and $PredA''$, we denote with S' and S'' the sets of the attributes for which the condition in the ancestor is relaxed with reference to that in $PredA'$ and $PredA''$, respectively.

Rule 2.1. Given two NGPSJ expressions e' and e'' and their ancestor $e' \oplus e''$, such that

$$(ExtP(e') = ExtP(e' \oplus e'') = ExtP(e'')) \wedge$$
$$(ExtS(e') = ExtS(e' \oplus e'') = ExtS(e''))$$

for each P', M', P'', M'' such that $\pi_{P',M'}(e')$ and $\pi_{P'',M''}(e'')$ are NGPSJ expressions in normal form, it is

$$\pi_{P',M'}(e') \oplus \pi_{P'',M''}(e'') = \pi_{P,M}(e' \oplus e'')$$

where $P = P' \overset{P}{\oplus} P'' \overset{P}{\oplus} S' \overset{P}{\oplus} S''$,

$$M = \begin{cases} M' \cup M'', & \text{if } P' = P = P'' \\ G(M') \cup M'', & \text{if } P' < P = P'' \\ G(M') \cup G(M''), & \text{if } P' < P \wedge P'' < P \end{cases}.$$

This rule can be applied when both source expressions are aligned to the ancestor as to their outer pattern and their selection predicates: in this case, a new aggregation level can be reached (see Figure 3). Pattern P cannot be coarser than S' and S'' to allow rewriting of predicates on attributes. As to aggregate measures, they must be transformed through function G when the outer pattern of the new ancestor, P, is finer than P' and/or P'': in fact, in this case, an intermediate aggregation level is being added and each aggregate function must be "decomposed".
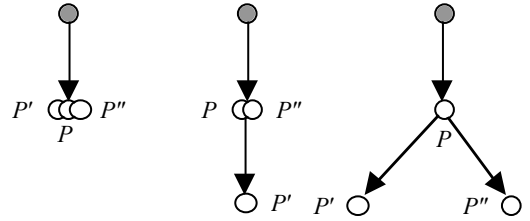

Figure 3. Different cases for Rule 2.1; the gray circles represent $ExtP(e') = ExtP(e' \oplus e'') = ExtP(e'')$.

Example 5. Let

$$e' = \sigma_{R>1000}\pi_{SName,TMonth,PName,N=MIN(Qty),R=SUM(Qty \cdot Prc)}$$
$$\sigma_{PCategory \ IN \ ('x','y','z') \ \wedge \ TYear<1999}(T)$$

$$e'' = \sigma_{R>1000}\pi_{SName,TMonth,PName,Q=SUM(Qty),R=SUM(Qty \cdot Prc)}\sigma_{PCategory='x'}(T)$$

$$e' \oplus e'' = \sigma_{R>1000}\pi_{SName,TMonth,PName,Q=SUM(Qty),N=MIN(Qty),R=SUM(Qty \cdot Prc)}$$
$$\sigma_{PCategory \ IN \ ('x','y','z')}(T)$$

$P' = \{TMonth, PType\}$, $P'' = \{TMonth, SCity, PName\}$,
$M' = \{AVG(N)\}$, $M'' = \{MIN(Q)\}$

It is

$$ExtP(e') = ExtP(e) = ExtP(e'') = \{SName, TMonth, PName\},$$
$$ExtS(e') = ExtS(e) = ExtS(e'') = (R>1000),$$
$$S' = \{TYear\}, \ S'' = \{PCategory\}$$

Since $P = \{TMonth, SCity, PName\}$, it is $P' < P = P''$, hence $M = \{AVG(N), COUNT(*), MIN(Q)\}$ and

$$\pi_{TMonth,PType,AVG(N)}(e') \oplus \pi_{TMonth,SCity,PName, \ MIN(Q)}(e'') =$$
$$\pi_{TMonth,SCity,PName,AVG(N),COUNT(*),MIN(Q)}(e' \oplus e'') ❏$$

Rule 2.2. Given two NGPSJ expressions e' and e'' and their ancestor $e' \oplus e''$, such that

$$(ExtP(e') = ExtP(e' \oplus e'') > ExtP(e'')) \wedge$$
$$(ExtS(e') = ExtS(e'') = TRUE)$$

for each P', M' such that $\pi_{P',M'}(e')$ is an NGPSJ expression in normal form, it is

$$\pi_{P',M}(e') \oplus e'' = \pi_{P,M}(e' \oplus e'')$$

where $P = P' \overset{P}{\oplus} ExtP(e'') \overset{P}{\oplus} S' \overset{P}{\oplus} S''$,

$$M = \begin{cases} M' \cup H(ExtM(e'')), & \text{if } P' = P = ExtP(e'') \\ G(M') \cup H(ExtM(e'')), & \text{if } P' < P = ExtP(e'') \\ M' \cup G(H(ExtM(e''))), & \text{if } P' = P \wedge ExtP(e'') < P \\ G(M') \cup G(H(ExtM(e''))) & \text{if } P' < P \wedge ExtP(e'') < P \end{cases}.$$

This rule can be applied when the outer pattern of one of the source expressions is coarser than that of the ancestor, meaning that an intermediate aggregation level has been previously added (see Figure 4). The aggregate measures in e'' must be transformed through H when the new ancestor is aligned with e'' (since each aggregate function must be "recomposed"), through $G(H)$ otherwise.
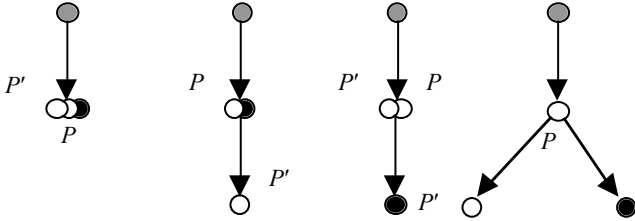

Figure 4. Different cases for Rule 2.2; the gray and the black circles represent $ExtP(e') = ExtP(e' \oplus e'')$ and $ExtP(e'')$, respectively.

Example 6. Let

$e' = \pi_{SName,TMonth,PName,Q=SUM(Qty)} \sigma_{PCategory='x'}(T)$

$e'' = \pi_{TMonth,PType,A=AVG(Qty)} \sigma_{PCategory\ IN\ ('x','y','z')\ \wedge\ TYear<1999}(T)$

$e' \oplus e'' = \pi_{SName,TMonth,PName,Q=SUM(Qty),A=AVG(Qty),C=COUNT(*)}$
$\sigma_{PCategory\ IN\ ('x','y','z')}(T)$

It is

$$ExtP(e')=ExtP(e)=\{SName,TMonth,PName\},$$
$$ExtP(e'')=\{TMonth,PType\},$$
$$ExtS(e')=ExtS(e)=ExtS(e'')=TRUE,\ S'=\{PCategory\},\ S''=\{TYear\}$$

and thus $\pi_{TYear,MIN(Q)}(e') \oplus e'' = \pi_{TMonth,SCity,PCategory,WAVG(A),MIN(Q)}(e' \oplus e'')$. ❏

Rule 2.3. Given two NGPSJ expressions e' and e'' and their ancestor $e' \oplus e''$, such that neither Rule 2.1 nor Rule 2.2 can be applied, it is

$$\pi_{P',M'}(e') \oplus e'' = e' \oplus e'',$$
$$e' \oplus \pi_{P'',M''}(e'') = e' \oplus e'',$$

$$\pi_{P',M'}(e') \oplus \pi_{P'',M''}(e'') = e' \oplus e'' \ .$$

5.3 Predicates on measures

Rule 3.1. Given two NGPSJ expressions e' and e'' and their ancestor $e' \oplus e''$, such that

$$(ExtP(e') = ExtP(e' \oplus e'') = ExtP(e'')) \wedge$$
$$(ExtS(e') = ExtS(e' \oplus e'') = ExtS(e'') = TRUE)$$

for each $PredM'$, $PredM''$ such that $\sigma_{PredM'}(e')$ and $\sigma_{PredM''}(e'')$ are NGPSJ expressions in normal form, it is

$$\sigma_{PredM'}(e') \oplus \sigma_{PredM''}(e'') = \sigma_{PredM'\ \vee\ PredM''}(e' \oplus e'')$$

(where either $PredM' \neq TRUE$ or $PredM'' \neq TRUE$)

A new selection predicate on measures can be added to the ancestor only if the source expressions are aligned to their ancestor in terms of outer pattern. Disjoining the source predicates allows rewritability to be preserved.

Example 7. Let

$e' = \pi_{SName,TMonth,PName,Q=SUM(Qty)}\ \sigma_{PCategory='x'}(T)$

$e'' = \pi_{SName,TMonth,PName,N=MIN(Qty)}\ \sigma_{PCategory\ IN\ ('x','y','z')\ \wedge\ TYear<1999}(T)$

$e' \oplus e'' = \pi_{SName,TMonth,PName,Q=SUM(Qty),N=MIN(Qty)}\ \sigma_{PCategory\ IN\ ('x','y','z')}(T)$

It is $ExtP(e')=ExtP(e)=ExtP(e'')$ and thus $\sigma_{N>100}(e') \oplus \sigma_{Q>1000}(e'')$
$= \sigma_{N>100\ \vee\ Q>1000}\ (e' \oplus e'')$. ❏

Rule 3.2. Given two NGPSJ expressions e' and e'' and their ancestor $e' \oplus e''$, such that Rule 3.1 cannot be applied, it is

$$\sigma_{PredM'}(e') \oplus \sigma_{PredM''}(e'') = e' \oplus e''$$

5.4 The ancestor-building algorithm

Comparing two NGPSJ expressions means determining their rewritability relationship; as already stated, this relationship can be inferred from their ancestor. The ancestor can be calculated by repeatedly applying the rules presented in the previous subsections, proceeding from inside the two expressions and, at each step, adding a new operator to the result.

The conditions placed in the head of the rules ensure that, at each step, exactly one rule can be applied; thus, the ancestor determined is unique. It should be noted that rules may leave the ancestor unchanged (for instance Rule 2.3, and Rule 3.1 if one of the two predicates is true). It is possible to verify that, in this case, any other rule further applied still leaves the ancestor unchanged. Thus, the ancestor-building procedure terminates when either no more rule can be applied (meaning that both source expressions have been completely processed) or the rule applied does not modify the ancestor.

Example 8. In the following we show how the ancestor between two sample expressions e' and e'' is computed.

$e' = \pi_{TYear,MAX(D)}\pi_{TMonth,SCity,PName,D=MIN(Q)}\sigma_{R>1000}$
$\pi_{SName,TMonth,PName,Q=SUM(Qty),R=SUM(Qty.Prc)}\sigma_{PCategory='x'}(T)$

$e'' = \pi_{TYear,AVG(N)}\sigma_{R>1000}\pi_{SName,TMonth,PName,N=MIN(Qty),R=SUM(Qty.Prc)}$
$\sigma_{PCategory\ IN\ ('x','y','z')\ \wedge\ TYear<1999}(T)$

STEP 1. By applying Rule 1:

$$\sigma_{\text{PCategory='x'}}(\mathsf{T}) \oplus \sigma_{\text{PCategory IN ('x','y','z')} \wedge \text{TYear}<1999}(\mathsf{T}) =$$
$$= \sigma_{\text{PCategory IN ('x','y','z')}}(\mathsf{T})$$

STEP 2. Let

$$e'_2 = \sigma_{\text{PCategory='x'}}(\mathsf{T}), \; e''_2 = \sigma_{\text{PCategory IN ('x','y','z')} \wedge \text{TYear}<1999}(\mathsf{T}),$$
$$e_2 = \sigma_{\text{PCategory IN ('x','y','z')}}(\mathsf{T})$$

By applying Rule 2.1:

$$\pi_{\text{SName,TMonth,PName,Q=SUM(Qty),R=SUM(Qty.Prc)}}(e'_2) \oplus$$
$$\pi_{\text{SName,TMonth,PName,N=MIN(Qty),R=SUM(Qty.Prc)}}(e''_2) =$$
$$= \pi_{\text{SName,TMonth,PName,Q=SUM(Qty),N=MIN(Qty),R=SUM(Qty.Prc)}}(e_2)$$

STEP 3. Let

$$e'_3 = \pi_{\text{SName,TMonth,PName,Q=SUM(Qty),R=SUM(Qty.Prc)}}(e'_2),$$
$$e''_3 = \pi_{\text{SName,TMonth,PName,N=MIN(Qty),R=SUM(Qty.Prc)}}(e''_2),$$
$$e_3 = \pi_{\text{SName,TMonth,PName,Q=SUM(Qty),N=MIN(Qty),R=SUM(Qty.Prc)}}(e_2)$$

By applying Rule 3.1:

$$\sigma_{\text{R}>1000}(e'_3) \oplus \sigma_{\text{R}>1000}(e''_3) = \sigma_{\text{R}>1000}(e_3)$$

STEP 4. Let

$$e'_4 = \sigma_{\text{R}>1000}(e'_3), \; e''_4 = \sigma_{\text{R}>1000}(e''_3), \; e_4 = \sigma_{\text{R}>1000}(e_3)$$

By applying Rule 2.1:

$$\pi_{\text{TMonth,SCity,PName,D=MIN(Q)}}(e'_4) \oplus \pi_{\text{TYear,AVG(N)}}(e''_4) =$$
$$= \pi_{\text{TMonth,SCity,PName,A=AVG(N),C=COUNT(*),N=MIN(Q)}}(e_4)$$

STEP 5. Let

$$e'_5 = \pi_{\text{TMonth,SCity,PName,D=MIN(Q)}}(e'_4), \; e''_5 = \pi_{\text{TYear,A=AVG(N)}}(e''_4),$$
$$e_5 = \pi_{\text{TMonth,SCity,PName,A=AVG(N),C=COUNT(*),N=MIN(Q)}}(e_4)$$

By applying Rule 2.2:

$$\pi_{\text{TYear,MAX(D)}}(e'_5) \oplus e''_5 = \pi_{\text{TYear,PCategory,MAX(N),WAVG(A,C)}}(e_5)$$

Since $e' \oplus e''$ is different from both e' and e'', the rewritability relationship between e' and e'' falls in case D (see Figure 2). ❏

6. CONCLUSIONS

In this paper we showed how to compare two NGPSJ expressions by determining an ancestor expression on which both can be rewritten, and we proposed a set of rules which recursively compute the ancestor.

NGPSJ expressions can be used to model expressively multidimensional queries; in this case, query rewriting is at the core of the optimization process. On the other hand, in [3] we showed that, in multidimensional databases, materializing summary views defined by NGPSJ expressions may lead to a relevant performance improvement as compared to materializing classical views as in [1]: on a TPC-D-based benchmark, and using a cost function which expresses the total number of disk pages which must be accessed in order to solve each query, the workload cost drops in the average to about 50%.

REFERENCES

[1] Baralis, E., Paraboschi, S., and Teniente, E. Materialized view selection in multidimensional database. In Proc. 23rd Int. Conf. on Very Large Data Bases (Athens, Greece, 1997), 156-165.

[2] Cohen, S., Nutt, W., and Serebrenik, A. Algorithms for rewriting aggregate queries using views. In Proc. Int. Workshop on Design and Management of Data Warehouses (Heidelberg, Germany, 1999).

[3] Golfarelli, M., and Rizzi, S. View Materialization for Nested GPSJ Queries. In Proc. DMDW'2000 (Stockholm, Sweden, 2000).

[4] Gray, J., Bosworth, A., Lyman, A., and Pirahesh, H. Data-Cube: a relational aggregation operator generalizing group-by, cross-tab and sub-totals. Technical Report MSR-TR-95-22, Microsoft Research, 1995.

[5] Gupta, A., Harinarayan, V., and Quass, D. Aggregate-query processing in data-warehousing environments. In Proc. 21st Int. Conf. on Very Large Data Bases (Zurich, Switzerland, 1995).

[6] Gupta, H., and Mumick, I.S. Selection of views to materialize under a maintenance cost constraint. In Proc. Int. Conf. on Database Theory (Jerusalem, Israel, 1999).

[7] Theodoratos, D., and Sellis, T. Data warehouse configuration. In Proc. 23rd Int. Conf. on Very Large Data Bases (Athens, Greece, 1997), 126-135.

[8] Yan, W.P., and Larson, P. Eager and lazy aggregation. In Proc. 21st Int. Conf. on Very Large Data Bases (Zurich, Switzerland, 1995), 345-357.

[9] Zaharioudakis, M., Cochrane, R., Lapis, G., Pirahesh, H., and Urata, M. Answering complex SQL queries using automatic summary tables. In Proc. ACM SIGMOD 2000 (Dallas, TX, 2000), 105-116.